Detection and Mitigation of GNSS Interference for Robust Navigation and Timing

Lloyd, Elizabeth

Award date:
2020

Awarding institution:
University of Bath

University of Bath

Department of Electrical and Electronic Engineering

A thesis submitted for the degree of Doctor of Philosophy

Detection and Mitigation of GNSS Interference for Robust Navigation and Timing

Author:
Elizabeth Lloyd

Supervisor:
Dr. R.J. Watson

Abstract

GNSS have been operational since the mid 1990s, and are now ubiquitous in modern life. Aside from the obvious positioning and navigation applications, it is also widely used for timing applications such as in communications and power distribution networks. The design of GNSS means the signals are weak, making them vulnerable to interference. This can be intentional or unintentional. Locating the source of intentional jamming is of particular interest as it is often associated with other criminal acts like car theft and drug dealing. Jamming detection and localisation systems already exist. However, none of those on the market today are low power and low cost, as well as capable of real time localisation of the jamming source, which is vital for mitigation of interference.

In this Thesis, the design, manufacture and real world testing of a system for GNSS jamming detection is demonstrated. The system relies on simple beamforming techniques, providing significant cost and power savings over the usual mathematical methods used for determining direction of arrival. To improve the efficiency of the search, bio-inspired algorithms are used. They are modified to allow tracking of a moving signal. Finally, they are implemented on an FPGA. The entire system, comprising antenna, digitally controlled beamformer, FPGA and real time data logging is demonstrated tracking a moving jammer in a real world scenario.

Contents

List of Figures

List of Tables

List of Abbreviations

Σ, Δ Sum, Difference

ADC Analogue to Digital Converter

AGC Automatic Gain Control

AIS Automatic Identification System

AJR Automatic Jamming Recognition

ALM Adaptive Logic Module

ALUT Adaptive Lookup Table

AoA Angle of Arrival

ANPR Automatic Number Plate Recognition

ASIC Application Specific Integrated Circuit

C/A Coarse Acquisition

C/N_0 Carrier to Noise Ratio

C-ITS Cooperative Intelligent Transport System

C4ADS Centre for Advanced Defence Studies

CAF Cross Ambiguity Function

COTS Commercial Off-The-Shelf

CPU Central Processing Unit

CRPA Controlled Radiation Pattern Antenna

CSV Comma Separated Variables

DAC Digital to Analogue Converter

DC Direct Current

DLR Dedicated Logic Register

DoA Direction of Arrival

DSP Digital Signal Processing

EBG Electronic Band Gap

EOC Early Operational Capability

EDA Electronic Design Automation

EPSRC Engineering and Physical Sciences Research Council

ES Electronic (warfare) Support

ESPRIT Estimation of Signal Parameters via Rotational Invariance Techniques

FFT Fast Fourier Transform

FLOPS Floating Point Operations Per Second

FPGA (-SoC) Field Programmable Gate Array (-System on a Chip)

FR4 Flame Retardant 4

GA Genetic Algorithm

GLA General Lighthouse Authority

GLONASS Globalnaya Navigazionnaya Sputnikovaya Sistema

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPU Graphics Processing Unit

GSM Global System for Mobile communications

HDL Hardware Description Language

HPBW Half Power Beam Width

I/O Input/Output

I/Q In phase/Quadrature

I^2C Inter-Integrated Circuit

IC Integrated Circuit

IP Intellectual Property

IWO Invasive Weed Optimisation

LFSR Linear Feedback Shift Register

LHCP Left Hand Circularly Polarised

LNA Low Noise Amplifier

LORAN LOng RAnge Navigation

LSB Least Significant Bits

LTE Long Term Evolution

LUT Look Up Table

MIPS Million Operations Per Second

MMIC Monolithic Microwave Integrated Circuit

MUSIC MUltiple SIgnal Classification

NP Non-deterministic Polynomial-time

PC Personal Computer

PCB Printed Circuit Board

PMU Phase Monitoring Unit

PNT Position, Navigation and Timing

PSO Particle Swarm Optimisation

RAIM Receiver Autonomous Integrity Monitoring

RAM Random Access Memory

RF Radio Frequency

RHCP Right Hand Circularly Polarised

RNG Random Number Generator

ROM Read Only Memory

RSS Received Signal Strength

RTOS Real Time Operating System

SA Simulated Annealing

SBAS Satellite Based Augmentation System

SBC Single Board Computer

SDR Software Defined Radio

SENTINEL GNSS SErvices Needing Trust in Navigation, Electronics, Location and timing

SMA Subminiature version A

SNR Signal to Noise Ratio

SPI Serial Peripheral Interface

SUV Sports Utility Vehicle

SV Space Vehicle

TDoA Time Difference of Arrival

UAV Unmanned Aerial Vehicle

UART Universal Asynchronous Receiver Transmitter

ULA Uniform Linear Array

URA Uniform Rectangular Array

USB Universal Serial Bus

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuit

VNA Vector Network Analyser

WWII World War 2

Chapter 1

Introduction

This Chapter aims to give the background and purpose of the work presented in this Thesis. A brief history of GNSS (Global Navigation Satellite Systems) and their forerunners is given. GNSS have many and varied uses in modern society, explaining why interference such as jamming and spoofing is such a concern. Different types of jammers, which are easily available on the market despite being illegal, are discussed. By considering how they work, it is possible to work out how to detect and mitigate them. An overview of existing (and proposed) jamming detection methods are presented, with a discussion of their benefits and shortcomings. The overall plan for the work in this Thesis is also laid out.

1.1 A history of electronic navigation

A basic timeline of key events in the history of electronic navigation is given in Fig. 1.1. The idea of electronic navigation systems that operate over a wide area has existed since before World War II. Early systems such as Gee and Loran-A used the principle of hyperbolic navigation, a concept first written about in the 1930s [1]. WWII precipitated a need for these systems and several were designed. The first to be built was Gee, which became operational in 1942 [2]. It was used to allow aircraft to navigate to targets and then return to base.

Gee used the time of flight of a signal from a pair of masts to determine a locus along which a receiver was located. Calculating the time of arrival draws circles around each mast. The radius of the circle is the distance from the mast as calculated using the equation Distance = Speed$\times$Time, where Speed is the speed of the signal's propagation and Time is the measured time of flight of the signal. Unfortunately this method required precise knowledge of the time the signal left the transmitter, which was not feasible. Instead the Time Difference of Arrival (TDoA) was used. For this method, the relative arrival time of two signals is calculated. If the difference between the times is zero, the receiver is equidistant between the two masts. The receiver will lie somewhere on a straight line that is perpendicular to a line

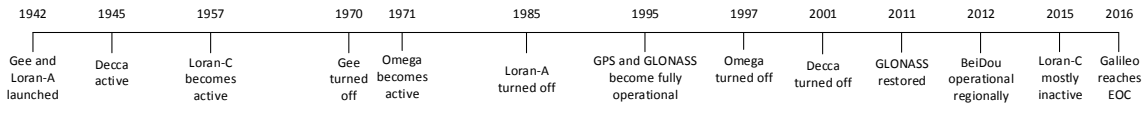


Figure 1.1: A timeline of various global navigation systems' operational periods.

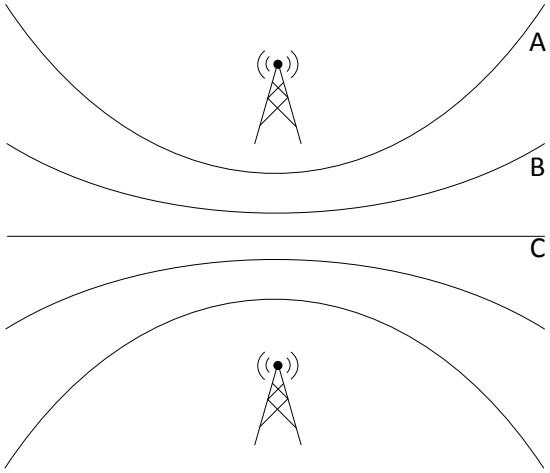


Figure 1.2: Hyperbolae drawn between one pair of masts.

joining the two masts (line C of Fig. 1.2). However, if the TDoA is non-zero then the straight line becomes a hyperbola (lines A and B in Fig. 1.2). Each point on this hyperbola is not equidistant from the pair of transmitters; rather, the hyperbola indicates all the positions in which one transmitter is a given distance further away than the other. Adding a second pair of masts allows calculation of a second hyperbola and the aircraft is found at the intersection of the two hyperbolae.

Gee had many advantages that placed it above other guided landing systems of the time. The most important for future navigation systems was the fact it was passive. This proved useful as it did not require aircraft to transmit and therefore give away their position. It also meant that the system was not targeting a certain user and all aircraft in an area could use it at the same time. This is just like the modern GNSS.

Gee inspired another system that became operational around the same time - the American Loran-A (originally just called LORAN, short for LOng RAnge Navigation). It operated at a lower frequency than Gee (1.75-1.95 MHz, as opposed to, originally, 20-30 MHz for Gee). This made the receivers larger, and it was mostly used for nautical purposes. It was less accurate but longer range as the lower frequency allowed the signal to bounce off the ionosphere. Loran-A stations, like those for Gee, were built in groups of transmitters (called 'chains'). In each chain would be one master and at least two slaves, spaced several hundred miles apart. The master would be triggered by an electronic clock to send its signal. Shortly afterwards a slave (equipped with a receiver) would receive that signal. When that happened, the slave would send its own signal. The stations were positioned so that the time between signals was 1 ms.

To simplify the receivers, in 1945 the Decca Navigator System was introduced. Instead of the pulsed signals that required Gee or LORAN receivers to calculate the time difference of arrival, Decca used the phase of the transmitted signal for navigation. It was mostly aimed at ships navigating in coastal waters around the UK. It was significantly more accurate than LORAN (accurate to within 200 m, compared to the tens of miles of Loran-A). Its accuracy and high levels of adoption meant it held off competition from other, much newer systems (such as Loran-C, whose first chain was built in 1957) until it was finally switched off in 1970.

The principle of operation of Decca was as follows. Transmission stations were organised into groups. Each group was assigned a base frequency and each station within the group would transmit on a harmonic of that frequency. This allowed the signals from different stations to be distinguishable by the receiver, but they could transmit phase-locked signals. Comparing the difference in phase from two stations gives a parabola of potential locations. Again, considering signals from several pairs of receivers will give an exact location. Accuracy depended on a number of factors but close to land where the signal was strongest, errors could be as low as a few metres.

In 1957, Loran-C became active. (Loran-B was an attempt to make a low frequency version that operated at 180 kHz. Experiments began in 1945 but it was shut down. Little information exists as to why.) Loran-C was designed to combat the problem encountered by previous systems, which found that it was not possible to make a system both long-range and highly accurate. By shaping the pulses, a matched filter could be used to more accurately align pulses from several masts and so reduce errors down to a few tens of metres.

Development on the American GPS (Global Positioning System) system began in 1975. What began as work on computing the location of satellites for the Seasat ocean monitoring project soon became a project to develop a global navigation system, with unencrypted signals and low hardware cost meaning it was widely accessible. While earlier ground-based navigation systems had unencrypted signals, the large receivers required made them unsuitable except for military and shipping use. For instance, the first widely used Loran-C receiver weighed 45 kg, and was prohibitively complicated for day-to-day use.

The first satellite-based navigation system, GPS has a number advantages over land-based systems. For one, the distance for all these systems is calculated based on time of flight. Previously this method would require the system to carry a very accurate measure of the time. For an error in position of less than 10 m, the error in the frequency source would have to be less than 30 ns or 30 ppb, meaning the receiver would need an atomic clock or similar. Atomic clocks cost tens of thousands of pounds and are large (typically they use 19" rack-mounted form factors). Earlier systems got around this requirement by instead using the time difference of arrival. The solution chosen for the GPS project was to require an extra satellite to gain a fix, with the fix contains an additional dimension. The user effectively calculates their position in x, y, z and also time. This achieves the aim of GPS: to be cheaply available worldwide.

A second improvement of GPS over other systems of the time is its higher frequency, at 1.575 GHz. This is a necessity because the ionosphere will reflect signals below a certain frequency (the critical frequency). There is no exact cut-off, but signals below 40 MHz are almost completely reflected. As the frequency of a signal increases, the less it is affected by the ionosphere. A higher frequency will suffer more attenuation with distance, increasing system costs. On the other hand, a smaller antenna is required, potentially making receivers smaller, cheaper and more portable.

The Russian GLONASS system began development at around the same time as GPS. It operates on a similar principle but with a few differences. The first is the frequency - GLONASS G1 is centred on 1.602 GHz. It also uses a different method of differentiating between satellites, or Space Vehicles (SV). While GPS uses pseudorandom codes, GLONASS

Table 1.1: Comparison of GNSS constellations.

Constellation	Nation	Operational as of 2019?	Freq. of main signal	Bandwidth of main signal
GPS	USA	Yes, global	L1, 1.57542 GHz	20 MHz
GLONASS	USSR	Yes, global	G1, 1.598-1.602 GHz	0.511 MHz
Galileo	Europe	Early capacity, global	E1, 1.57542 GHz	24.552 MHz
BeiDou	P.R.C.	Regional only	B1, 1.5611 GHz	4.092 MHz

uses frequency division multiplexing so each satellite operates on a different centre frequency within the band. Another key difference between GPS and GLONASS is the orbit. While GPS orbits at 55°, GLONASS orbits at approximately 65°, making it slightly more accurate at higher latitudes.

Galileo is a European project. It is currently in Early Operational Capability (EOC), able to supply a weak signal to receivers, with full service expected in 2020. The main signal, called E1, is centred on 1.57542 GHz, the same as GPS. As a result receivers do not need dual band or wide band antennas, keeping costs down. The E1 band is slightly wider than GPS's L1, and the accuracy of the system is planned to be slightly higher.

BeiDou, also known as Compass, is the Chinese GNSS system. Its satellite constellation has a different structure to other GNSS. The constellation is a successor to the regional BeiDou-1 constellation and so BeiDou-2 (as 'Compass' is more correctly called) will have a number of geostationary satellites for backwards compatibility. Global coverage will be provided by 27 medium Earth orbit and three inclined geosynchronous satellites.

Table 1.1 gives a summary of the different constellations and the frequency of their main (unencrypted) signal. The most obvious takeaway from this information is that, no matter how old the constellation (GPS was first designed in the 1970s, while Galileo was conceived as a modern system with the latest technology), they all have centre frequencies within just a few tens of MHz. While this means a receiver can be easily designed as the antenna only needs to cover a small band, it also means interference can relatively easily affect reception of all constellations, not just one. As will be discussed in Section 1.3, for all their differences, the systems have the same weaknesses.

1.2 Uses of GNSS

GNSS are most well known for their use in in-car sat-nav systems. These first came into use in the early 2000s, during a period when, for political reasons, GLONASS was not fully operational. As a result GPS became the most commonly used system, although modern receivers are now often multiconstellation. Typically receivers are able to receive GPS but will use GLONASS (most commonly) for additional accuracy, and perhaps also Galileo. Some are even able to receive BeiDou but this varies by region. It has been recognised [3] that a loss of GNSS would cause an economic loss due to increased traffic and longer journey times as drivers may struggle with navigation, as well as hindering the emergency services. However, an outage would affect many other aspects of modern life.

GNSS-derived positions are used by trains. If there was a GNSS outage the capacity of the rail network would be reduced as the locations of trains would be less certain and so

bigger safety margins would be required. Airports also use GNSS to assist with landing of aircraft, particularly in bad weather. By leaving a receiver in a single location for a long period of time, an average position can be calculated. This position is likely to be extremely accurate as the averaging will have smoothed out any random variations. Aircraft landing at the airport can then use this very precise position, as well as any calculated corrections, to land with a lower chance of go-arounds. GNSS interference or outages would remove this additional information, resulting in increasing costs and lower capacity.

In shipping, modern navigation is aided by AIS (Automatic Identification System, an inter-ship communications system that reports position, course and speed to aid navigation). This system gathers information on position, speed and heading and transmits it to all nearby ships, allowing them to easily avoid each other. The position information is derived from GNSS so a disruption would result in ships needing to travel slower and employ more crew to watch for ships and prevent collisions. The result would be higher transport costs.

GNSS is mostly known for its positioning and navigation. However, the decision to design the system so that the user does not require an accurate time source, and instead to allow that information to be extracted from the GNSS signal itself, has had additional benefits. In fact, it is estimated that only 10% of the one billion GNSS receivers worldwide are used for positioning [4]. The remainder are used for timing purposes. Many systems require an accurate time signal to work without glitches or errors, and GNSS provide a (relatively) low cost, infrastructure-free method of disseminating time to even remote locations. While caesium frequency sources (and other atomic clocks) could be used, they have a tendency to drift. GNSS are often used to tune atomic clocks and prevent this drift, leaving the clock as a backup.

GNSS time is used in mobile networks. LTE ('4G') networks require synchronisation of 16 ppb to prevent calls and data packets from dropping and to allow adjacent base stations to provide efficient service at the edge of cells [5]. If service was disrupted it would cause disruption to anything that relied on the mobile network, as not just 4G but also 2G GSM and 3G rely on good synchronisation.

Timing is also used in banking. Transactions need to be timestamped to ensure they happen in the correct order to prevent erroneous charges. This includes personal banking but also in stock exchanges, where automated transactions can be happening thousands of times per second. If GNSS was disrupted this could cause problems such as flash crashes, where the stock market can very rapidly lose a lot of value, often before quickly recovering. This could potentially cost huge amounts of money.

Another important use of GNSS based timing is in power distribution networks. With the increase of renewable energy and its tendency to produce power with erratic frequencies and phases, it is vital that the grid remains stable across a country, or even further afield. GNSS-derived timing allows for extremely precise tracking of frequency and phase, making grids more efficient and stable. This use will only increase as smart grids become the norm. A GNSS outage would cause generators to become out of sync. There are already systems in place in the UK to prevent this happening and if the frequency deviates too far from what is acceptable, whole areas can have their power shut off, as happened in August 2019 [6]. In this instance the cause was two malfunctioning power generation plants, resulting in a hospital

losing power and hours of delays on trains.

1.3 GNSS interference

If GNSS are useful in so many applications, the question arises why people would want to jam it. The answer tends to be criminal activity. Often, jammers are used in car theft. Modern high value vehicles such as SUVs and construction plant are common targets for theft and so are fitted with GPS trackers to assist with recovery after theft. Using a jammer allows a thief to steal the car and defeat the tracker [7]. Jammers can assist taxi drivers in fraudulently increasing their earnings. By jamming their GPS trackers, a taxi can appear to be in a different location and so can collect fares that might otherwise be assigned to other drivers [8]. A final example of GPS jammers being used to evade authorities is described in [9]. While the article recommends against it, it does mention that an ankle bracelet for monitoring of individuals under house arrest or similar conditions could be bypassed using a GPS jammer.

All satellite-based navigation systems suffer from the same weaknesses to interference. The carrier frequency of GPS is at approximately 1.575 GHz, which has a wavelength of around 19 cm. This high frequency is chosen to allow signals to propagate through the ionosphere. The ionosphere attenuates low frequency signals (below approximately 20 MHz) significantly more than high frequency signals, so the GPS signal is able to pass through relatively unhindered. Using a high frequency means that the attenuation between transmitter and receiver is increased; the signals are strong when they leave the satellite but after travelling 20,180 km (for GPS) the signal level can be as low as -130 dBm at the receiver. However, as a good point to balance the increased attenuation, smaller antennas are possible. It means that receivers can be made smaller, but also means that transmitters such as jammers can also be made smaller. The low power and small antenna size mean that even small battery powered jammers can easily overwhelm a nearby receiver. They are also low cost and easily concealed.

All GNSS jammers work in a similar way. By producing a signal in the frequency band of interest, they 'drown out' the legitimate satellite signal in the surrounding area. The area of effect varies depending on the output of the jammer. A survey of jammers available on the market was conducted in [10]. A number of different designs and styles were tested. Some were battery powered and some were powered from the cigarette lighter socket of a vehicle. These cigarette lighter style jammers are common due to the use of low cost jammers as 'privacy protection devices', marketed as allowing a driver of a fleet car to avoid tracking. An example is shown in Fig. 1.3. Other common jammers are battery powered. Some have external antennas (*e.g.* Fig. 1.4), while others have internal antennas. A typical 'cigarette lighter' style jammer has an output of around 50 mW, while a battery powered jammer could be 300 mW or more.

Humphreys *et al.* tested 18 commercially available jammers [10]. All had signals with frequency modulation. The main reason for this design choice is most likely to be manufacturing cost: by adding a frequency sweep, the system can be designed less precisely as it is more likely that at least some of the band of interest will be jammed. However, despite

Figure 1.3: Cigarette lighter type jammer. Seized by police in Scotland.

Figure 1.4: Battery powered jammer with external antenna.

this large target it was found that some jammers (two of the 18 tested) did not produce any frequencies within the 20 MHz wide L1 band. Others did not jam 1.575 GHz but did cover at least some of the band.

Some of the jammers had unusual sweeps. Some had artefacts overlaid on the frequency sweep, meaning it was not consistent over time. It is not known the effect this would have on the jammer's ability to deny GPS. This, coupled with the fact that some jammers failed to jam the L1 band, lends credence to the theory that jammers are made cheaply.

It has been noted that with all jammers tested at the University of Bath, the jammers have non-directional antennas. The jammers studied do not intend to target a single receiver and instead simply deny GPS to an area. The signals are also always on - there is no pulse or sweeping of the beam. All of these features must be considered when designing a detection and location system, and will be discussed again in Chapter 2.

Sometimes jamming can be unintentional. Typically this occurs when a GNSS antenna malfunctions due to poor installation or wear and tear, and begins re-radiating. For instance, a GPS antenna at a telecoms operator's site was found to be suffering from interference from a nearby antenna [11]. A similar event happened at a data centre [12], which was found to be caused by a poor connection inside a 20 year old GPS receiver nearby.

Unintentional jammers are unlikely to have the same frequency characteristics as a device made for jamming. The signal might be continuous wave, emitting at the resonant frequency of the receive antenna. The signal may be intermittent depending on the cause; however, it will be unlikely to have evenly spaced pulses. The radiation pattern and any sweeping of the beam will correspond to the design of the 'rogue' antenna.

While unintentional jamming does happen, the majority of events are deliberately caused by jammers. There are many reasons that people may want to jam GNSS. Often the devices are advertised as having a range of only a few metres. They are bought by those wishing to avoid being tracked while driving, for instance, company-owned fleet vehicles [13]. When Humphreys *et al.* measured the output power of jammers [10], it was calculated that some jammers could prevent acquisition as far as 8.5 km away in the most extreme case. A small privacy protection device was the cause of repeated daily outages at Newark Liberty airport [14, 15] in 2009. After months of investigation it was found that a van driver using a nearby road was using a small jammer for privacy protection. The van driver was inadvertently disrupting an entire airport. In addition, these events are not rare with on average 4.8 alerts per day between 2013 and 2017 detected at an airport [15].

The effect of jammers on consumer hardware has also been documented. It may seem obvious, but the work in this Thesis would have little use if it turned out that receivers were not actually disrupted by interference. Fortunately (for the purposes of this research) it has been found that interference does impact receivers. A very strong jammer will deny GPS to a receiver and cause it to move to back up methods, if available. However, a weaker signal (perhaps encountered at the edge of the range of a jammer) may not be detected [16]. The receiver may not lose lock but will receive a degraded signal, which affects the accuracy of the position (and timing) calculation. Even receivers that can use multiple constellations will suffer as jammers are wideband [17], although it was observed that a Galileo receiver was able to track at a slightly lower carrier-to-noise (C/N_0) ratio than a GPS receiver.

Many instances of jamming have been detected and reported upon. While hyperbole can make a problem look much more serious than it is, there is evidence that the concern is warranted. Reports have been commissioned by the governments in both the UK and the USA on the impact of jammers. The report commissioned by the UK, [18], calculated that the cost to the UK economy of a five-day outage of GPS would be £5 bn. Similarly in the US it is estimated that since GPS first became available in the 1980s, it has added $1.4 tn to the US economy [19]. Other reports have been published by specific industries. For example, in 2013 Symmetricom published a white paper on the effect of jamming and spoofing on power distribution networks [20]. This report identified the risks from both reradiating antennas and drivers wishing to avoid tracking. In Australia the problem of taxi drivers jamming GPS became so prevalent, the police in Melbourne carried out an operation to educate them on the law and the possible impact of jammers [21].

The SENTINEL (SErvices Needing Trust in Navigation, Electronics, Location and timing) project came about as a result of concern over jamming in the UK [15]. It a Chronos Technology project funded by Innovate UK and EPSRC. It placed a network of detectors around the UK, including by airports and ports. The project was helped establish the scale of jamming in the UK, but it also provided real-time information on jamming events.

In 2017 it was reported that at one airport alone, in the four years between 2013 and 2017, there were on average 4.8 jamming events per day detected by a SENTINEL GPS monitoring system. In a city environment 5732 GPS jamming events were detected, adding up to 4.6 days of outages, with one event lasting 60 minutes [22]. The overall trend is that the number of jamming events is increasing.

In addition to jamming, there is the problem of spoofing. Jamming is a simple problem, requiring the emitter to simply drown out the legitimate GPS signals to the point where the receiver can no longer determine its position. Spoofing is the broadcasting of a fake GPS signal to deceive a receiver. This is an altogether more complicated matter as most receivers are resistant to sudden large changes in apparent time or position and will simply treat the spoofing as additional noise. To spoof effectively, the attacker must first determine the position of the target receiver and broadcast a convincing GPS signal. The signal power is increased until the receiver begins to use only the spoofed signal, and only then can the apparent position of the receiver be gradually moved away [23].

Despite it being a complicated process, there have been examples of spoofing being carried out successfully. The most famous is Humphreys attacking the autopilot of a yacht in 2013

[24]. By spoofing GPS signals, the autopilot of a yacht was fooled into directing the ship off course. Since then cases have been observed in the Black Sea near Russia, and near the port of Shanghai [25]. In the most recent case, a ship that was berthed was reported to be travelling at 7 knots in the channel. Over a period of minutes it appeared on AIS systems to be berthed, then moving at various speeds, then berthed again. This pattern was identified by the UK General Lighthouse Authority (GLA) as characteristic of GPS interference during trials in 2009 [26]. The Centre for Advanced Defence Studies (C4ADS) published a report in 2019 on GPS spoofing by Russia and Syria [27]. They documented a number of incidents where ships at sea appeared to be inland, at an airport.

Both of these spoofing attacks, in Shanghai and in Russia, are suspected to have been from nation states; so far no small, low cost spoofing systems have been widely reported to be creating attacks. This is not to say that they will not in the future, especially as a Software-Defined Radio (SDR) can now be used to create a spoofer for less than $350 [27]. In 2016, D. Schmidt *et al.* correctly predicted that shipping would be targeted by spoofing attacks [28], which is exactly what has been shown. They also predict that spoofing threatens electric smart grids, criminal tags (ankle bracelets) and perhaps even trains. Symmetricom's white paper [20] on the vulnerabilities of the power grid to GPS interference agreed with this assessment. While it covered many types of vulnerabilities (including jamming as well as plain old equipment failures), they noted that it has been proved that a spoofing attack can cause Phase Monitoring Units (PMU) to malfunction. An experiment in [29] demonstrated that in just 20 minutes a PMU can be spoofed to read a phase angle 70° from the true value, which is considered to be a large difference.

1.4 Existing detection and mitigation technology

Jamming is an increasing problem, and so systems are being developed to detect and mitigate it. Some systems attempt to detect the interference and perhaps revert to a backup system. Others use modified antennas to reduce the probability that a system is jammed in the first place. A third class of system does not attempt to mitigate the jamming and simply detects and logs events, potentially allowing for other nearby systems to be protected. This third type also aid law enforcement agencies in tracing criminals who use jammers.

There are multiple ways of detecting the presence of jamming. Some methods attempt to detect a change in the C/N_0 ratio of the incoming signal, with an increase in noise indicating the presence of a jammer. Others look for a sudden change in the overall signal strength, indicating that a new emitter on the GPS frequency has suddenly appeared within range of the receiver.

The u-blox NEO-6 claims to have defence against jamming, although no description of the method is given. However, in tests it was found that the spurious continuous-wave signal produced by an antenna resonating would effectively jam the receiver [30]. At the same time the receiver reported a very low probability of jamming. This small sample size test would imply that some devices may not be as effective as they claim. u-blox devices are generally relatively low-value, aimed at consumers, and so the design is likely to be less robust. If the system is more safety-critical, better detection algorithms are required.

Tani *et al.* proposed a jamming detection method suitable for safety of life applications [31]. They also give a more thorough discussion of methods for detecting jamming. The methods can be split into two groups: pre-correlation and post-correlation. Precorrelation detection looks to detect the jamming prior to searching for satellites, while post-correlation monitors parameters such as the C/N_0 for evidence of interference. Post-correlation methods generally perform worse as they rely on the true signal being accurate. If the jamming is low level, gradual degradation of the signal may prevent the jamming from being detected.

The method presented in [31] requires multiple Fast Fourier Transform (FFT)-like operations to be performed on the signal prior to it entering the GPS decoder. These operations provide information on the incident signal and it can be decided if the signal environment is normal or if there is evidence of interference. This could be carried out using a modified front end and would be able to be retrofitted to existing systems. However, in practice this would have high computational demand and therefore high power consumption (practicalities of the implementation were not considered in the paper and results were from simulations only).

Once the jamming has been detected the system could revert to a backup to avoid disruption. The option proposed in [32] was to use Network Time Protocol, but this requires a network connection which may not be available in all situations. Other backups include atomic clocks with sufficient stability to maintain service until the GNSS signal has returned. The exact situation will determine what exactly counts as 'sufficient' stability. A caesium frequency standard is a commonly used backup, but these are expensive and have high power consumption - typically on the order of 50 W [33]. Finally, none of these backups are useful if the receiver is being used for positioning or navigation rather than timing.

A less common solution is to use a Controlled Radiation Pattern Antenna (CRPA) to screen the receiver from the interfering signal. This would require the addition of a front end that can quickly determine the direction of the jamming, then modify the beam pattern so as to exclude the interfering signal. It would be important to maintain the overall shape of the beam pattern to allow the GPS signal to remain strong. A method for this was proposed by Haupt *et al.* in [34]. By only allowing the algorithm to optimise the least significant bits of the phase setting, nulls could be moved around to point towards interferers without significantly changing the main beam direction or shape. In this example the challenge is then to determine the direction of arrival of the interference so that it can be blocked.

The first type of system tried to detect the interference so that it could be mitigated. The second type of systems do not detect jamming, but aim to be more resilient to it so it is less likely to have an effect. Many lower cost systems are in this category. Tests carried out by Chronos Technology Ltd. [35] used two u-blox M8 front ends, each with a different Tallysman antenna. Both antennas were multi-constellation, capable of receiving GPS, GLONASS, Galileo and BeiDou, as well as Satellite-Based Augmentation System (SBAS) signals. However, one was a filtered antenna, model number TW3712, that claimed to reduce interference from harmonics and nearby frequencies. However, in tests it was found that over the course of the day, the system with the filtered antenna lost its fix for more time than the unfiltered antenna, indicating the effectiveness of these types of systems may be limited.

Rather than using software or filtering to reduce the probability of jamming, assumptions can be made about the jamming and the receiver antenna can be designed to limit the impact

of interference based on these assumptions. For example in a typical scenario, the receiver is looking towards the sky for the GNSS satellites. However, jammers are typically on the ground, in the plane of or perhaps even below the receiver antenna. A solution to this would be a choke-ring antenna [36], which aims to maximise the front-to-back ratio thus reducing the impact a jammer behind or below the antenna could have.

The third group of jamming detection systems are not attempting to protect themselves, but instead aim to allow jammers to be tracked and traced. This is of particular interest to law enforcement agencies, as often jammers are used to aid in the committing of other crime. For example, the SENTINEL system mentioned earlier allowed jamming events to be tracked.

Chronos Technology Ltd produce a range of handheld jammer detectors. The simplest, the CTL3510, simply reports the presence of jamming [37]. The more advanced CTL3520 [38] can display the signal strength as the user moves the unit around. These systems are small, low cost, light weight and battery powered, making them practical in the field. Their disadvantage is the lack of automation in the decision making process, requiring a human operator to determine the actual direction of the jammer.

In-car GNSS jammers can interfere with smart transport systems [39]. It was proposed that they could be detected by performing an FFT on the received signal. However, it was only tested experimentally, and would require a wide network of vehicles that used the (at the time, proposed only) Cooperative Intelligent Transport System (C-ITS), an ad-hoc peer-to-peer communications network between vehicles. The disadvantage of this method of tracking and locating of jammers is the requirement for a large network of sensors and a central data centre to establish which vehicle is jamming.

In a similar attempt to track vehicles carrying jammers, Kar *et al.* in 2014 proposed and simulated a network of static and mobile sensors to track a vehicle with a jammer across a road network [40]. The static sensors would detect jamming near critical infrastructure. The energy level of the GPS band was monitored and when it increased past a certain point, an alarm was triggered and sent back to a data centre. This theory behind this part of the system is that as GPS signals are so weak (typically below the noise floor), any detectable energy in that band must be interference. In addition to the static sensors, mobile monitors would detect anomalous C/N_0 values. By noting all the times that jamming was detected, linking this with Automated Number Plate Recognition (ANPR) technology, and assuming that all vehicles travel at slightly different speeds and will eventually diverge, the exact source of the jamming can be determined. As above, a large network of sensors was required for this proposed network.

Released a short time later, the CTL3530 [41], also known as Jammercam, or AJR (Automatic Jamming Recognition), uses a modified CTL3520 to capture a photograph of a vehicle with a jammer as it passes the unit. It is also capable of sending the image on to a server to allow action to be taken. The system is not able to discern between different vehicles on a multi-lane motorway so this is best deployed on areas like entrance and exit ramps to motorways. However, a single unit is capable of identifying a particular vehicle, unlike other proposed roadside systems.

The idea of using a network of small sensors has been proposed multiple times for tracking

vehicles. More generally, Lindström *et al.* used an ASIC (Application Specific Integrated Circuit) to power a small sensor capable of detecting jamming on L1 and E1, the main GPS and Galileo frequency bands. By measuring the gain setting of an AGC (Automatic Gain Controller), it can be established when jamming is taking place; in the presence of high signal levels the gain will reduce, indicating that a jammer is nearby [42].

While the Chronos series of devices fulfil the requirements of being small and low cost and do not require a large network of sensors, they have weaknesses. The CTL3510 and CTL3520 are handheld, meaning by necessity they can only operate at hand height. This becomes a problem in highly scattering environments such as in container storage yards at ports. This very specific example is used because of the prevalence of GPS jamming during vehicle theft. High value vehicles are stolen and often shipped abroad, but before they leave the country they are likely to spend an amount of time being stored among other containers. At shoulder height there is significant amounts of scattering due to the relatively flat walls of shipping containers. Moving the detection system above the corridors may reduce the multipath and make the Line-of-Sight signal clearer.

If a jammer detection system was to be flown above the ground, there are two options. A helicopter would be prohibitively expensive for most law enforcement agencies, leaving an Unmanned Aerial Vehicle (UAV) or 'drone' as the better choice. However, the 'unmanned' element of this solution means the detection system must be automated and able to determine for itself the location of the jamming.

1.5 Research description

The aim of this work is to investigate a number of areas related to low power, low cost detection, location and mitigation of jamming and other GPS interference. This is a research project and as such the end goal is not a single polished product; instead, it is an investigation and demonstration of new technology that could be incorporated into complete systems in the future. Potential uses include a UAV-mounted system for locating jamming in highly scattering environments, or as part of a jamming mitigation system.

The research discussed in this Thesis attempts to answer a number of questions:

1. Is it possible to detect, locate, and/or mitigate GNSS jamming using a low power, low cost, automated system?

2. Is it possible to find out information about the jammer in question?

3. What methods can be used to speed up the rate at which it locates the jamming, without significantly increasing the power consumption of the system?

Chapter 2 describes the design of an antenna suitable for locating GPS interference. It examines the literature for up-to-date methods of localisation and design accordingly. The key components of the design are its ability to accurately locate a jamming source with minimal power and cost, answering question 1. above.

Chapter 3 builds on the work in Chapter 2. The optimum design was shown to be an array antenna, and so in Chapter 3 an element with which to build this array is described.

This element is small and lightweight, made from cutting edge, high quality materials. It also has the ability to measure the polarisation of the incoming signal, which may provide information about the interference source (question 2.).

Chapter 4 describes the design of a modular high frequency beamformer. This beamformer allows the testing of different antenna and array designs with the lowest cost possible as it can be reconfigured for different designs. The design uses Commercial Off-The-Shelf (COTS) components, rather than custom designed ones, chosen for their low cost and low power consumption, answering question 1. In addition, this Chapter demonstrates that system as described so far is capable of detecting and locating a GPS jammer without a human operator.

Chapter 5 focuses on software instead of hardware. The system built in Chapters 2 to 4 is capable of locating jamming using a slow and inefficient exhaustive search. In this Chapter a number of optimisation algorithms are considered as methods to increase the speed at which a jammer is found without fundamentally changing the method of searching or the hardware (question 3.). While none of the algorithms are novel in themselves, they have not been applied to emitter localisation before.

Chapters 6 and 7 look at improvements to the search algorithms. This includes modifying the algorithms in novel ways so that they are more suited to the task, and changing the platform on which they are run. This should both improve the speed with which the jammer is found, and also reduce the power consumption of the system (criteria 1. and 3.).

Finally in Chapter 8 conclusions will be drawn and future work will be discussed.

1.6 Contributions of this Thesis

A number of publications were written based on the work that is described in this Thesis. A paper on the array antenna that was designed in Chapter 2 was published and presented at the European Conference on Antennas and Propagation (EuCAP) in London in 2018 [43]. The beamforming network designed in Chapter 4 was presented at Loughborough Antennas and Propagation Conference (LAPC) in 2017 [44]. The use of bio-inspired algorithms to identify the strongest signal using the array and beamforming network was presented at EuCAP 2019 in Krakow [45], and the initial findings regarding tracking moving jammers was presented at the virtual EuCAP 2020 [46].

The novel parts of this work are the use of bio-inspired algorithms to track moving emitters. Bio-inspired algorithms have been used for many tasks but not for jammer localisation. Additionally, the modifications to the bio-inspired algorithms to allow them to produce a solution every iteration, and the ability to react to a moving target, are entirely new. Finally, the invasive weed optimisation algorithm has not been implemented on an FPGA before.

Chapter 2

Design of an antenna for direction finding

In which: Localisation methods are reviewed . . . The system requirements are discussed . . . A method is chosen . . . A number of array topologies are considered . . . An array design is chosen

In Chapter 1, a variety of jamming mitigation techniques were discussed. It was observed that many of the methods relied on detecting the jamming and reverting to a backup such as an atomic frequency standard or network distributed time. A network connection may not be feasible in a remote location, while an atomic clock is large, heavy and consumes a lot of power. As such there may be instances where neither of these options are suitable. An alternative method is to determine the location of the emitter and use that information to either modify the receiver's beam pattern to exclude interference, or to find and neutralise the source of the interference and thus make GNSS based PNT (Position, Navigation and Timing) more reliable.

There are multiple ways of determining the position of an emitter. Some of them require the emitter to have certain properties, such as a known output or a pulsed signal. A variety of methods were considered and the best option chosen. Factors such as the power consumption and size of the system were considered, as the aim is to create a low power, low cost solution to detect and mitigate GNSS interference. The best option of those considered was to calculate the Angle of Arrival (AoA) as this could be used for mitigation (hardening a receiver against interference) or used to locate the emitter, and thus provide flexibility. It was also possible to perform AoA calculations with potentially a much lower power consumption than that required for other methods.

Once a method was chosen, a suitable antenna was designed. Three different designs were simulated, and the best one manufactured and tested. It was found that the simplest design was the most suited to jammer localisation, and the remainder of the work described in this Thesis is based upon this array design.

2.1 Background

To deal with jammers and other interferers, first they must be located. There are many different methods for locating a transmitter as it has been a problem to solve since radio communications were first used. This Section will study the literature for suitable methods.

2.1.1 Time Difference of Arrival

In Chapter 1 it was explained how a receiver can use the Time Difference of Arrival (TDoA) of two signals from two different transmitters to find its position. The inverse can also be performed: by correlating the same pulse at two different receivers, a hyperbola of possible transmitter positions can be drawn.

Typically the distance between the transmitter and a receiver, $d_{\text{rec}_1,\text{trans}}$, is calculated using (2.1), where c is the speed of light. This calculation finds the time of flight, by finding the difference between when the signal arrived at the receiver (t_{rec_1}) and when it left the transmitter (t_{trans}), and hence find the distance between the transmitter and the receiver.

$$d_{\text{rec}_1,\text{trans}} = c \times (t_{\text{rec}_1} - t_{\text{trans}}) \tag{2.1}$$

It will be very difficult, if not impossible, to determine the time of transmission from an uncooperative transmitter. As a result the time of arrival method cannot be used to locate jammers. Hence the time of flight from the transmitter to two different receivers is considered. By subtracting one from the other the time *difference* of arrival is calculated (2.2).

$$\Delta d_{\text{rec}_{1,2},\text{trans}} = c \times ((t_{\text{rec}_1} - t_{\text{trans}}) - (t_{\text{rec}_2} - t_{\text{trans}})) \tag{2.2}$$

Assuming the two pulses being measured are the same pulse, the time of transmission t_{trans} will be the same for both signals and so is cancelled out (2.3).

$$\Delta d_{\text{rec}_{1,2},\text{trans}} = c \times (t_{\text{rec}_2} - t_{\text{rec}_1}) \tag{2.3}$$

This method of locating an emitter was studied in depth by Okello *et al.* in a series of papers. The first paper in the series [47] simulated using a swarm of UAVs. Each was capable of detecting a pulsed signal, time stamping it accurately, and reporting the information back to a 'fusion centre'. In 2019 a 'swarm of UAVs' typically refers to a number of small quad-copter type units, each weighing 1 kg at most. However the research assumed the platforms would be sufficiently large that they could also carry a precision clock and other Electronic warfare Support (ES) sensors. It was also assumed that the platforms would be fixed-wing types which must keep moving to stay aloft, which would not be necessary with multirotor type UAVs. However, at the time of writing of Okello's first paper in the series in 2006, DJI, who now have a 70% market share for multirotor UAVs (which are capable of hovering), had only just been formed.

Simulations showed that a swarm of three UAVs was able to locate an emitter. However, there were a number of problems and assumptions. The work assumed that the UAVs were positioned close enough together that there was no ambiguity when associating pulses. It assumed the emitter was located on the ground (which is a reasonable assumption) and

produced very poor estimates of altitude when a three dimensional problem was considered. It also assumed there was no noise. There were other problems with the work. For instance, the communications bandwidth requirements are very high. Additionally, there would be significant computational load to calculate optimum trajectories for the UAVs, although this could be solved by using multirotor type UAVs instead of the fixed wing type that were available at the time.

The next paper in the series [48] sought to overcome the bandwidth problem by only flying two UAVs. The system has high bandwidth requirements because for each pulse that arrives at each UAV must be timestamped and reported back, so limiting the number of UAVs reduces the bandwidth requirement. However, for a two-dimensional fix on the position of an emitter, a minimum of three UAVs are needed, to generate three hyperbolae and thus calculate the position. To reduce the number of UAVs and thus decrease the bandwidth, several measurements were taken over time as the UAVs moved, and the position detected after the fact, once the data has been combined at the fusion centre. This has the disadvantage of increasing the minimum time required for a position fix.

Limiting the number of UAVs not only reduces the bandwidth but also simplifies measurement association and path planning. 'Measurement association' refers to the work performed by the fusion centre. Each UAV will be receiving pulses, time stamping them and reporting them. However, the system as a whole has no way of knowing that two pulses seen by two different receivers are the same pulse from the transmitter. The solution was found using Kalman filters [49], showing that it is possible to pinpoint an emitter's location even if the measurements are taken over a long period of time. The Kalman filter works by assuming that both UAVs are illuminated by the transmitter's beam at the same time. It also assumes that the leading edge of a pulse is accurately detected and timestamped. A timestamped pulse at one UAV will be selected and the pulses received by the other UAV are compared to find which is the most likely to correspond. Pairs of measurements taken over time can be plotted and a solution found. The paper obtained good results but relied on two rather large assumptions. The first is that the system communication bandwidth was unlimited and UAVs would be able to report on every pulse received, which is likely not the case. The second assumption, in order to gain accurate results, is that there is a third UAV, which runs counter to the aim of the work which was to limit the bandwidth by only using two UAVs. This third UAV must be positioned carefully to improve the results, although the ideal geometry is not described in the paper.

To increase the efficiency of the measurement association, a number of recursive algorithms were then compared [50]. It is at this point that the researchers admit that their fastest option takes more than two minutes to calculate an answer. To mitigate an interference event the system needs to be able to react quickly. At an airport the mean duration of a jamming event was just 17 seconds [22]. Hence a faster solution is important.

Aside from the fact the system was slow, there are multiple other reasons why this particular method is not suitable. The basic premise is that the system correlates pulses from an emitter and uses them to determine a time difference of arrival. Unfortunately GPS jammers do not have a pulsed signal. It may be possible to still use this method if the signal had a known output and a matched filter could be used to identify a common point in the signal.

Propagation time could then be determined. However, this would require a known output. As every jammer has a different output due to manufacturing differences this could not be known *a priori* and so would have to be calculated for each jamming event. There is the added complication that some jammers do not have a frequency chirp and are continuous wave, so this method would not work at all for some types of interference.

Another problem with this method is the requirement for a high accuracy clock for time-stamping of signals. An error of just 33.3 ns would result in an error of 10 m, so the clock would have to be very low error. The solution would probably be an atomic frequency standard (as GNSS are likely to be unavailable during a search for a GNSS jammer) which would be too large, heavy and power hungry to fly on a modern multirotor UAV. The aim of this Thesis is to create a small, low cost and low power method of detecting and mitigating jamming to avoid needing expensive backup systems like atomic clocks. Hence requiring one to mitigate the interference defeats the purpose.

Okello *et al.* only provided simulations for this research. In contrast Bhatti *et al.* demonstrated a working, TDoA-based, GPS jammer localisation system [51]. The system used GNSS and signals of opportunity (such as mobile phone signals) to gather the timing information. To localise the interference source it used two receivers: one was fixed and one was mobile (this time based in a car, not a UAV), allowing multiple hyperbolae to be drawn. The system did locate the jammer with good precision, but there are a number of caveats. Firstly, it is recognised by Bhatti that the accuracy of TDoA suffers if the emitter's frequency spectrum is not flat, which cannot be assumed when the emitter is an unknown quantity. Second, they assumed that there was prior knowledge about the properties of the signal: in this case they created the 'jammer' themselves, using a frequency in the US amateur radio licence band to allow for open-air testing. The signal produced by the jammer mimicked the GPS L1 C/A code (the freely available civilian code), which has good properties regarding autocorrelation. This will have aided the system in associating measurements. The paper freely admits that this is not possible in a real scenario. The system demonstrated in this paper was also very large, taking up the front seat of a passenger car. The final problem with the system is that while it successfully located the jammer with a good degree of accuracy, it was certainly not in real time. Instead the data was recorded then processed using MATLAB after-the-fact. The time required to produce a result is not stated. All of this leads to the conclusion that the localisation of a real jammer in real time has not yet been managed with a small, low power system.

2.1.2 Scan-based

TDoA is a common method for locating transmitters, but other researchers have proposed alternatives. The scan-based method proposed by Hmam in [52] does not require the signal to be pulsed, making it more suitable for use with GNSS interference. Once again the receivers are carried on UAVs, indicating that UAV-based location methods are worthy of investigation.

In this research by Hmam, a swarm of UAVs would record the time at which a swept signal passed each UAV; the time at which the signal strength peaks is recorded. Assuming the sweep rate is constant, the angle between the emitter and the two receivers can be established, and therefore the location of the transmitter can be calculated. This idea was proposed to

overcome the requirement for highly accurate, synchronised clocks, which were required for TDoA measurements. The method is much less sensitive to clock errors, requiring an accuracy on the order of microseconds. In contrast TDoA requires nanosecond accuracy. Once again this method requires a swarm of UAVs as each measurement requires two physically separate points. It also assumes that the beam is swept, and only has one main beam. This is unlikely to be true for a GPS jammer, which will broadcast in all directions, or a spoofer, which may broadcast indiscriminately or may target a particular receiver but will not sweep. In the case of unintentional interference there is a chance the emitter will sweep, depending on the exact failure mode, but this is a niche case. As a result, scan-based emitter localisation is not an option. However, once again UAVs are considered a suitable platform.

2.1.3 Received Signal Strength

Instead of relying on the signal to have certain features such as a pulse, or the beam to be swept, it is possible to determine the location based on a feature every single emitted signal has: power. In [53], Liang *et al.* use the Received Signal Strength (RSS) of a signal to determine its location. The measurements are performed using a UAV. RSS is related to the distance from the transmitter but also the transmitter output power. To combat this ambiguity, a fixed receiver is also placed. The relative signal strength at one of the mobile receivers relative to that at the fixed receiver gives a direction and distance in which to search. As there is no requirement for correlating signals between multiple UAVs, the hardware is much smaller and lower cost.

The paper presented the results of simulations only, and made a number of assumptions. The key one is that all propagation encountered follows the log-normal model. This is generally considered to be accurate but cannot take into account very local effects that may confound it, such as shadowing from buildings, without modelling the environment beforehand. The system was also very sensitive to noise, particularly at lower frequencies. It would not be able to detect multiple emitters so despite its fast operation and relatively simple hardware requirements, this method is not suitable.

2.1.4 Angle of Arrival

The final common method for localisation is Angle of Arrival (AoA), sometimes also called Direction of Arrival (DoA). This aims to measure the direction of the strongest signal(s); multiple measurements from different positions allow the location of the emitter to be determined. This is separate from TDoA, which calculated a hyperbola of potential positions but has no knowledge of where the signal originated. It is also different to the scan-based and RSS methods, which only considered the total received signal strength.

Fig. 2.1 gives a general overview of the different approaches for calculating DoA. One method is to use classical techniques such as beamforming. For instance, a phased array or parabolic dish antenna can be used to produce a narrow beam. The beam is then swept over the search space and when the received signal is at a maximum, the direction has been determined. This method has the disadvantage that a very large antenna is needed to produce a sufficiently narrow beam (more detail on the reasons for this are found in Section 2.2.3.1). As

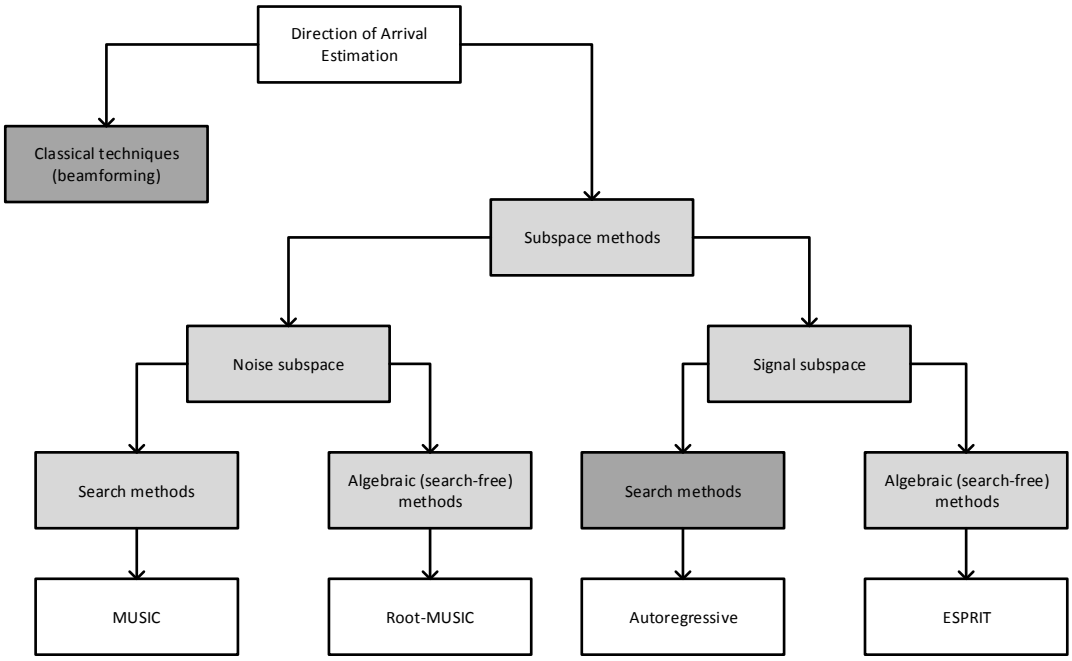


Figure 2.1: An overview of direction of arrival methods. Adapted from [54]

nulls are sharper than beams, another common method is to use the null of a loop antenna to search for signals. When the received signal strength is at a minimum, the null of the antenna beam pattern is pointing at the signal source [55]. However, the search speed of this method is limited by the speed at which the antenna can rotate mechanically. In contrast, an electronically steerable phased array could be moved by any angle in a time on the order of 1 ms.

Subspace methods rely on algorithms to use information about signals arriving at an array of antennas to determine the DoA, among other things. Only some subspace techniques will be discussed here: methods that search the signal subspace are not common compared to the MUSIC and ESPRIT algorithms and their variations, so they will be neglected. Fig. 2.1 mentions them purely for completeness. One of the common DoA algorithms is MUSIC (Multiple Signal Classification). It was first proposed by Schmidt [56] as an algorithm that can be used to estimate parameters of an incoming signal. In this situation only the DoA part of the algorithm would be used.

MUSIC is based upon recording signals impinging on an array [57]. Referring to Fig. 2.2, there are M sources and N receivers. The m^{th} signal source transmits a signal α_m. The signal received at each antenna array element x is the sum of all these signals, modified by the propagation environment, plus some noise, n, (2.4).

$$\mathbf{x} = \sum_{m=1}^{M} \alpha_m \mathbf{s}(\phi_m) + \mathbf{n} \tag{2.4}$$

This assumes that the array is a uniform linear array (ULA). In this equation, α_m describes the propagation between the signal sources and the receiver, where the attenuation is a and the phase shift is b, as in (2.5).

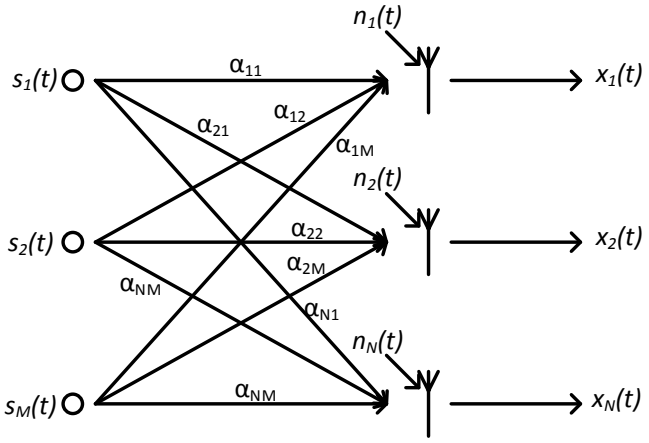


Figure 2.2: Generic system diagram showing M signals impinging on an array antenna with N elements. Adapted from [59].

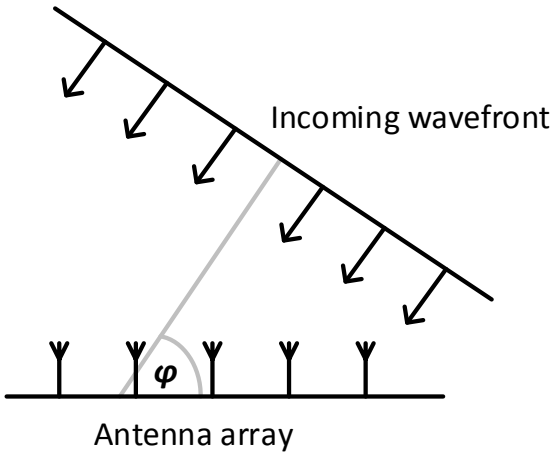


Figure 2.3: A single plane wave incident on an array antenna.

$$\alpha = ae^{jb} \tag{2.5}$$

The first stage of MUSIC requires the population of an $N \times M$ matrix of steering vectors, $\mathbf{S}$. This matrix comprises a number of 'snapshots', which are signal steering vectors $\mathbf{s}(\phi)$ in (2.6) (adapted from [58]). The signals in this steering vector are recorded so that both the magnitude and phase of the incident signal are known. They provide a mapping between the phase of the incident signal and the position of the antenna elements.

Fig. 2.3 shows one incoming wavefront (from a single source) incident on an array. The wave is assumed to be planar, meaning the receiver is in the far field. As the signal is not broadside to the array (*i.e.* $\phi \neq 90$), there will be a phase difference in the received signal at each separate antenna. Information regarding the separation between antennas is also included. As a result, the antenna array must be well characterised beforehand.

$$\mathbf{s}(\phi) = \begin{bmatrix} \exp\left(-j\frac{N-1}{2}\frac{2\pi}{\lambda}d_x \sin\phi\right) \\ \exp\left(-j\frac{N-1}{2}\frac{2\pi}{\lambda}d_x \sin\phi\right) \\ \vdots \\ \exp\left(j\frac{N-1}{2}\frac{2\pi}{\lambda}d_x \sin\phi\right) \end{bmatrix} \tag{2.6}$$

For each 'snapshot', an estimated autocorrelation matrix is calculated by multiplying the steering vector with the Hermitian transform of itself. A number of snapshots are taken and the autocorrelation calculated for each. The mean of K estimated autocorrelation matrices can be found, giving a good approximation R for the true autocorrelation matrix, as in (2.7).

$$\mathbf{R} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k\mathbf{x}_k^H \tag{2.7}$$

Each correlation matrix estimate ($\mathbf{x}_k\mathbf{x}_k^H$) will have a number of correlation peaks, some corresponding to signals and some to noise. If the noise is uncorrelated, when they are averaged the noise will disappear and only peaks corresponding to signals will remain. Over a sufficiently large number of snapshots (generally, $K > 2N$) it is assumed that the estimated correlation matrix will converge towards a good approximation of the true correlation matrix. This assumption only works if the noise is uncorrelated. If the noise is correlated, such as in the presence of multipath, spurious peaks will be present and the correlation of the true peaks will be reduced.

Once a good approximation of the correlation matrix has been estimated, the eigenvectors of that matrix can be calculated. They are calculated such that the correlation matrix multiplied by an eigenvector is zero, making the eigenvectors ($\mathbf{q}_m$) orthogonal to the signals of interest.

$$\mathbf{R}\mathbf{q}_m = 0 \tag{2.8}$$

The eigenvectors can be combined into a matrix, $\mathbf{Q}n$, where n refers to the fact these are noise eigenvectors. They can be used to plot the MUSIC 'pseudospectrum'.

$$P_{MUSIC}(\phi) = \frac{1}{\mathbf{s}^H(\phi)\mathbf{Q}_n\mathbf{Q}_n^H\mathbf{s}(\phi)} \tag{2.9}$$

The steering vector is multiplied by the matrix of eigenvectors and plotted against the direction of arrival, ϕ. The output is zero when in the direction of a signal, and non-zero elsewhere. Plotting the reciprocal gives large peaks in the directions of signal sources, which are narrow and therefore easy to identify. The magnitude of the peak indicates the signal strength and the amount of correlation in a given direction. Hence in a noisy environment the peak heights will be reduced as the correlation is lower. The narrow peaks mean MUSIC has good resolution compared to similar techniques (such as a periodogram).

MUSIC is not a search-free technique: this pseudospectrum must be searched to find the peaks. In addition, in a noisy environment the magnitude of signal peaks will be reduced while noise peaks will increase. The algorithm must know how many signals there are to find, else it will return as many as are asked for. This is a key weakness of this type of algorithm, as multipath would result in an unpredictable number of peaks.

Root-MUSIC is an adaptation of the MUSIC algorithm [60]. It performs a Z-transform on the output of MUSIC:

$$z = e^{jkd\cos\phi} \tag{2.10}$$

$$\mathbf{s}(\phi) = \left[1, z, z^2, \ldots, z^{N-1}\right]^T \tag{2.11}$$

$$\implies \mathbf{q}_m^H \mathbf{s} = \sum_{n=0}^{N-1} q_{mn}^* z^n = q_m(z) \tag{2.12}$$

Thus the eigenvectors have now been moved to the Z domain. They can be plotted on a pole-zero diagram. In an ideal, noiseless case all signal zeroes would be on the unit circle; in the presence of noise they will move away from the circle slightly. Zeroes caused by noise will be far away from the unit circle, possibly outside it, and will have images, making them easy to identify as noise so that they can be neglected.

Other variants of MUSIC attempt to deal with some of the limitations. For example, smooth-MUSIC splits the N elements into multiple overlapping sub-arrays and hence can remove the assumption that all incoming signals are uncorrelated. Other methods seek to make the antenna physically smaller, at the expense of some additional computation [61, 62].

ESPRIT (Estimation of Signal Parameters via Rotational Invariance Techniques) is an alternative to MUSIC [63]. ESPRIT, like root-MUSIC, is a search-free technique. While MUSIC and its derivatives are noise subspace techniques (because the eigenvectors mean the signal is non-zero when there is noise), ESPRIT uses the signal subspace to determine the angle of arrival. In ESPRIT, the steering matrix $\mathbf{S}$ is split into two sub-arrays, $\mathbf{S}_0$ and $\mathbf{S}_1$. They are related as in (2.13).

$$\mathbf{S}_1 = \mathbf{S}_0 \mathbf{\Phi} \tag{2.13}$$

Unfortunately the steering matrix $\mathbf{S}$ is not known and must be estimated. ESPRIT uses techniques such as total least squares (or any other optimisation technique) to find an estimate for $\mathbf{\Phi}$ and therefore the the eigenvalues and the directions of arrival. This algorithm is less reliant on the exact layout of the array than MUSIC, and has a lower computational complexity, but at the cost of a larger array.

A fundamental problem with all these subspace methods is their high power consumption. Much of this comes from the computational complexity, and can be reduced by choosing a suitable method (for instance, ESPRIT has lower computational cost compared to MUSIC, at the expense of a larger array). One aspect that cannot be changed, however, is the requirement to record the signals at the start of the process. For every element in the array, both In-phase and Quadrature (I and Q) signals must be recorded, and with sufficient resolution to capture the frequency and phase. This amount of hardware will have a high power consumption and cost. This alone makes algorithm-based DoA methods unsuitable for this research.

A third method, which bears more similarities to beamforming methods than to algorithm-

based methods, is switched beam systems. The basic idea is that an antenna array produces a number of beams to search in azimuth, with the calculated AoA being the direction with the strongest RSS. The many beams require hardware to produce, similar to a phased array, and the result can be sensitive to noise. Badawy *et al.* proposed a two-step method in [64]. The first step was to use a single element of the array to determine the presence of a signal. If a signal was detected the switched beam system was used. Instead of choosing the beam with the strongest signal, the result of each beam was cross-correlated with the originally detected signal. This more complicated method resulted in a performance similar to that of MUSIC. However, unlike an RSS-based system, this proposed method requires recording of signals for cross-correlation and so the power consumption will increase considerably.

One final method for determining AoA that was highlighted during the course of this research, but does not fit into one of the above categories, is modulated backscatter. Fusco *et al.* created a self-aligning wireless link using this principle [65]. This method can theoretically be used to ensure the reliability of a connection from an unstable platform such as a hovering UAV. It is achieved using the following method: a signal is transmitted at a target. The target modulates the amplitude of the signal as it is backscattered. The original transmitter receives the modulated signal as it returns from the target. By finding the conjugate of the received signal, the transmitting beam can then be pointed directly at the target. Separate transmit and receive channels would allow this process to be carried out continuously, keeping the target in the main beam of the transmitter. However, this method would not work in this scenario for a number of reasons, the main one being that the target must be cooperative. It also uses coherent detection, which requires prior knowledge of the signal structure. Finally it requires downconversion of the signal's frequency, which consumes a lot of power.

Having reviewed the literature to learn about common methods of emitter localisation, it is clear there is only one method that can reliably detect a GNSS jammer without requiring significant power: A *beamsteering* approach does not require the jammer to have any particular signal characteristics, does not require any prior knowledge and does not require power-hungry recording of the signal. It also only requires one platform, which reduces the cost of the system. Another advantage is the applicability to mitigation. By finding the angle of arrival of a signal, the impact of that signal on a receiver can be minimised without the emitter being located.

2.2 Specifications for design

When designing the array, it is important to identify what metrics will be used to declare if a design is 'good' or 'bad', as what works for one situation will not be useful for another.

2.2.1 Physical attributes

The antenna must be as small and lightweight as possible.

Ideally the antenna would be small and lightweight. This will make it more suitable for mounting on a UAV, one of the potential use cases of the technology developed in this project. If it is lighter, a smaller (and less powerful) UAV can be used. This will probably lower the cost of the system, in addition to the expected lower manufacturing cost from using

less material for the antenna. In addition, a smaller array will be less affected by factors like wind, making a UAV-mounted platform more stable and making results more accurate.

Making antennas physically smaller is always a topic of research, as many antennas have physical constraints on their size and shape. However, simply moving the elements closer together increases the coupling between elements. This coupling can introduce distortions in the beam pattern, as well as making the antenna less efficient. Electromagnetic Band-Gap (EBG) structures, commonly referred to as 'mushrooms', between elements can reduce coupling [66]. The mushroom name comes from the 'T'-shape of the structures, with a thin upright post and a flat top. They reduce mutual coupling by disrupting surface waves. As a result an antenna array can be fabricated with a smaller physical size while maintaining some of the benefits of a larger array. Techniques such as EBG structures may need to be employed but they will significantly increase the manufacturing cost because the fabrication is a non-standard process.

2.2.2 Power consumption

Power consumption should be minimised.

The antenna itself will not consume any power. However, as the antenna is a phased array with digital control to allow beamsteering, a larger antenna will require more components in the beamformer. The exact layout of the antenna may result in more components being added, hence increasing power consumption.

In order to make this system as light as possible, power consumption must be minimised so that as few batteries as possible are needed. This is the main driver for using the method of searching described, instead of the matrix methods, which require digitisation of the signal and significant computation which have high power consumption. However, even though this method should be lower power computationally, it is important to also minimise the power used by the beamformer.

The beamformers for different designs will probably all use similar components, so the power consumption will depend on the exact design and the power consumption can be qualitatively compared without specific devices being chosen beforehand.

2.2.3 Beam pattern

The beam pattern must give the system a good angular resolution to allow it to locate emitters with a high degree of precision. It must also be steerable over a wide angle.

A wide scanning angle is important for a UAV mounted system as demonstrated by Fig. 2.4. If the system can scan over a wide angle, a large area of ground can be searched without moving the UAV. A narrower beam will illuminate a smaller area of the ground and more scans would be needed to cover the same area. As the beam can scan on the order of milliseconds, but moving the UAV takes time on the order of tens of seconds, a wider scanning area will significantly increase the speed of locating jammers. This in turn reduces the cost and weight of the system as batteries do not need to last as long so can be smaller and therefore lighter and cheaper.

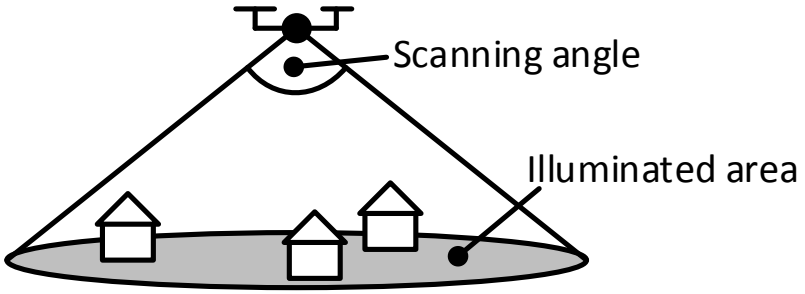


Figure 2.4: A demonstration of how the scanning angle affects the area that can be searched.

The shape of the beam at the edges of the illuminated area is also important. The ideal beam pattern (produced by considering the array factor only) has no sidelobes in the Σ pattern. However, this may not be the case once the real antenna elements are simulated. Larger sidelobes take power away from the main beam and so reduce the system's sensitivity. It would also reduce the angle over which the beam can be swept.

2.2.3.1 Antenna array theory

To understand how to design an antenna array with a suitable beam pattern, it is necessary to understand the design factors that can be changed and what effect those changes will have.

The directivity of an antenna is the measure of how focused the beam pattern is, compared to an isotropic antenna. The gain is the directivity multiplied by the efficiency of the system, including resistive losses and other factors that cause loss.

The gain of a system is related to the radiation efficiency and the aperture efficiency. The radiation efficiency is determined by the hardware of the system and takes into account the conductivity of the materials and so on. The aperture efficiency is the ratio of the effective aperture to the physical aperture. The physical aperture is simply the actual size of the antenna and is often written A_p. The effective aperture (A_e) is a measure of how much power from an incident plane wave is captured and delivered. There may be losses from fringing effects and the like, so that the effective aperture is less than the physical aperture. Hence the aperture efficiency $\eta_A = A_e/A_p$ is less than one.

It is generally true for all electromagnetic transmitters and receivers, be they telescopes, microscopes or antennas, that the half-power beamwidth (HPBW) can be roughly estimated based on the size of the aperture (D) and the frequency (and therefore wavelength, λ), as in (2.14), although the value of 1.27 depends on the aperture efficiency of the antenna.

$$\text{HPBW} = 1.27\frac{\lambda}{D} \tag{2.14}$$

This assumes that the aperture has been tapered to achieve a compromise between the narrowest possible main beam and the best sidelobe levels [67]. The implications of this equation are that a smaller antenna creates a wider beam. To improve the resolution of the system, the antenna can be made bigger. For an array antenna such as the one that will be designed in this Thesis, there are two ways of doing this: increase the inter-element spacing, or increase the number of elements.

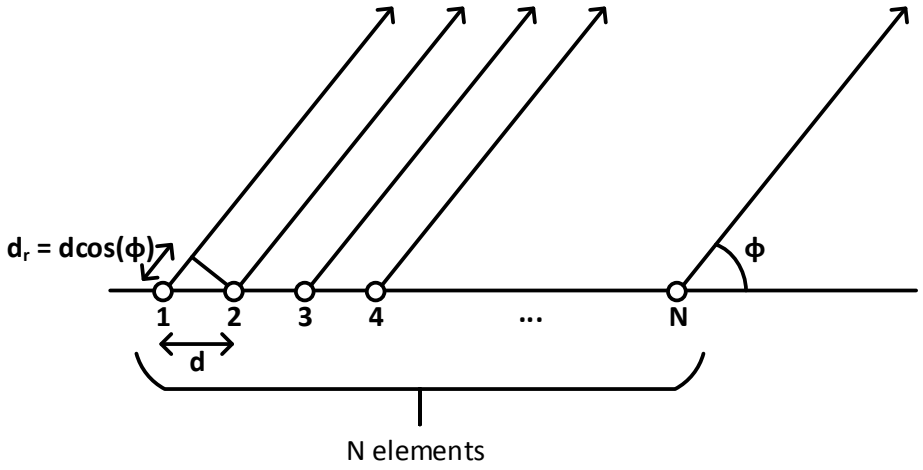


Figure 2.5: A theoretical uniform linear array of point sources.

Fig. 2.5 shows a hypothetical uniform linear array. There are N elements with the same inter-element separation of d. The angle between endfire and the beam direction of interest is ϕ. It is assumed that the target is in the far field. In the far field it can be assumed that a wavefront is flat. The far field is defined as when the target is more than $2D^2/\lambda$ away from the antenna, where D is the overall aperture size of the antenna and λ is the wavelength of the signal. It is also assumed that the difference in path length between the target and two different elements of the array has no effect on the amplitude of the signal. However, the phase change over the different path length is significant. The excess path length d_r is calculated in (2.15).

$$d_r = d\cos(\phi) \tag{2.15}$$

The phase change over this excess path, θ, is calculated in (2.16), where $k_0 = 2\pi/\lambda$.

$$\theta = k_0 d_r = k_0 d\cos(\phi) \tag{2.16}$$

An additional phase shift can be added between elements by using a phase shifter. This additional phase shift is denoted α, as per (2.17).

$$\theta = k_0 d\cos(\phi) \pm \alpha \tag{2.17}$$

If all the elements of the antenna array are identical, the resultant field E_R can be found by summing the individual contributions (2.18). The field resulting from element n is named E_n, with element 1 used as a reference.

$$E_R = E_1 + E_2\exp(j\theta) + E_3\exp(j2\theta) + \ldots + E_N\exp(j[N-1]\theta) \tag{2.18}$$

Assuming the amplitudes of all the elements are the same, this can be simplified as in (2.19).

$$E_R = E_0[1 + \exp(j\theta) + \exp(j2\theta) + \ldots + \exp(j[N-1]\theta)] \tag{2.19}$$

From (2.19) it is possible to derive an equation for the array factor and phase centre of the array. By analysing the array factor the direction of any beams and nulls can be established, which is vital for this research. The first step is to manipulate (2.19) into (2.21) using the identity (2.20), where x is substituted for exp and k is substituted for $jn\theta$.

$$\sum_{k=0}^{K} x^k = \frac{1 - x^{K+1}}{1 - x} \tag{2.20}$$

$$E_R = E_0 \left(\frac{1 - \exp(jN\theta)}{1 - \exp(j\theta)} \right) \tag{2.21}$$

This can be split into magnitude and phase terms (2.22). The magnitude term is the array factor, which is the relevant part in this case. The array factor describes how the radiated field changes with changing θ, d, N, α, and the frequency. The phase term gives the phase centre of the array. The phase centre is the point from which the radiation spreads out spherically.

$$E_R = E_0 \left| \frac{\sin\left(\frac{N\theta}{2}\right)}{\sin\left(\frac{\theta}{2}\right)} \right| \exp\left[j \left(\frac{(N-1)\theta}{2} \right) \right] \tag{2.22}$$

The overall meaning of this equation is that the resultant field in a given direction is a function of all of the individual components, and is affected by the phase shift between the elements (θ). It is a sinusoidal function; the modulus operation creates a power spectrum with only positive excursions. However, at the zero 'crossings' (which are now turning points) there will still be a null. As this is a sinusoidal shaped function the slope is at its steepest at these points, making the nulls very sharp. This is a key part of this research.

The array factor can be inspected to determine where the major lobes of the beam pattern will occur. As this is the magnitude term, signal power is at a maximum when the denominator is zero, *i.e.*, $\sin(\theta/2)$. This occurs when $\theta = 0$, 2π, ..., $2m\pi$, where m is an integer.

The angles of the main lobes are given by (2.23).

$$\theta = 2m\pi = k_0 d \cos(\phi_m) \pm \alpha \tag{2.23}$$

Rearranging to make the angle the subject gives (2.24).

$$\phi_m = \cos^{-1}\left(\frac{2m\pi \mp \alpha}{k_0 d} \right) = \cos^{-1}\left(\frac{m\lambda}{d} \mp \frac{\alpha}{k_0 d} \right) \tag{2.24}$$

Setting $m = 0$, for the first (main) beam gives (2.25).

$$\cos(\phi_0) = \frac{\alpha}{k_0 d} \tag{2.25}$$

This equation makes it clear that the direction of the main beam is linked to the phase shift between elements, α. If the phase shift between elements is zero, the main lobe is broadside to the array. If $\alpha = k_0 d$, the beam is endfire. For an array with an inter-element separation of $\lambda/2$, $k_0 d = \pi$. As π radians is equal to 180°, adding the elements in antiphase will generate nulls where previously there were peaks.

To decrease the beamwidth of the main beam, and thus increase the resolution of the system, the aperture size should be increased as per (2.14)[1]. However, if the inter-element spacing is increased, grating lobes will appear. Grating lobes are defined as lobes other than the main lobe, that have the same power level as the main lobe. Grating lobes would cause ambiguity in the direction of the signal, and so are undesirable. They would also reduce the overall gain of the system by reducing the amount of radiated power focused in the main beam. To find out how to prevent grating lobes, we return to (2.24). To find the location of the main beam, m was set to 0. To find the requirements for the first grating lobe, set $m = 1$. Substituting (2.25) into (2.24) gives (2.26).

$$\cos(\phi_m) = \frac{\lambda}{d} + \cos(\phi_0) \tag{2.26}$$

If the right hand side of the equation is forced to be greater than one the grating lobes will disappear. This is because to find the angle the inverse cosine must be calculated, and the inverse cosine of a number greater than one is complex and so the beam does not exist physically. Of the variables in (2.26), the only one that can be varied is the inter-element separation, d. (2.27) rearranges (2.26) to make d the subject.

$$d < \frac{m\lambda}{1 - \cos(\phi_0)} \tag{2.27}$$

Hence for an array where the main lobe is broadside ($\cos(\phi_0) = 0$), the separation between elements is $d < \lambda$ for grating lobes to occur. Typically separations of $\lambda/2$ are used because if the beam is steered away from broadside, the denominator becomes smaller as the $\cos(\phi_0)$ term grows, and grating lobes would begin to appear if a marginal case like $d = \lambda$ was used.

An interesting observation is that if the angle of the main lobe $\phi_0 = 90°$, that is the beam is endfire, no amount of reducing the inter-element separation will prevent grating lobes because the denominator would go to one. As the grating lobes are mirrored around broadside, there is guaranteed to be a second main lobe endfire to the array in the opposite direction. In between the two will be a null broadside to the array, which is the same as was observed if $\alpha = k_0 d$.

To prevent grating lobes the inter-element separation must be small, around $\lambda/2$. To make the aperture larger, therefore, more elements must be added to improve the beamwidth. More elements will result in a heavier antenna with higher power consumption by the beamformer. Thus a large array is not a solution.

The amplitude of each element in the array can also be set individually. A uniform distribution gives the highest gain and narrowest beam, also gives very high sidelobe levels, which could cause ambiguity in the result. A cosine or triangular amplitude shaping, for example, reduces the sidelobes, at the expense of making the main beam wider. It also reduces the overall efficiency of the antenna, η_A and does not prevent grating lobes. This method will also only work with arrays with sufficient elements to have shaping - a two element array cannot be shaped without becoming lopsided or simply reducing the overall gain of the antenna. Amplitude shaping may also require additional hardware in the beamformer (see

[1]Note that this is not necessarily the case with superdirectivity [68]. However, superdirective antennas are not suitable in this work as they are more complex than the solution chosen, and will not be considered.

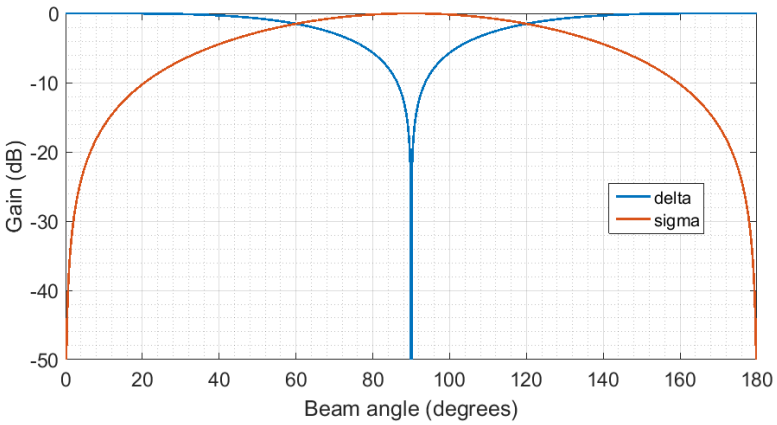


Figure 2.6: The theoretical output of a two element array either summed in phase or in anti-phase, based on array factor analysis of a pair of point sources.

Chapter 4 for a more detailed discussion).

The proposed method of improving the resolution is to use the null signal instead of the main beam. This method is inspired by the Chronos CTL3520 jamming detection and localisation system.

Summing the signals from the all of the elements in the array, with no additional phase shifts, gives a very wide beam at all points in front of the array; the outputs from the two antennas are effectively working together. This sum beam will be referred to as Σ in this work.

However, if the array is split in half, and one set of signals are added to the other set in antiphase, the two sets of signals will completely cancel out broadside to the array (again, assuming there are no additional phase shifts). This creates a null. Nulls are much sharper than peaks due to the fact they are created by, in effect, taking the absolute magnitude of a sinusoidal wave. At the point the wave crosses zero the gradient is at a maximum. This zero crossing is the location of a null. The null beam will be referred to as Δ. The antiphase sum is equivalent to subtracting one set of signals from the other, and is carried out by applying a 180° phase shift to one set of signals.

Both Σ and Δ are shown in Fig. 2.6. This graph demonstrates narrowness of the null compared to a main beam. The wide Σ beam which has low resolution, as predicted by (2.14). It also shows the Δ beam, with its very narrow and very deep null. Moving away from broadside will make the gain of the Δ beam rapidly increase, until it is almost the same as the Σ beam. As this is a graph of an array factor analysis and the y-axis is logarithmic, the null is infinitely deep. This will not be the case with simulations of real antennas.

If the null is being used for direction finding, instead of the main beam, then the direction of the jammer will be indicated by when the RSS is at a minimum. This is similar to using a mechanically steered loop antenna. However, as mentioned by Badawi *et al.* in [64], it is necessary to verify that an interference signal is indeed present. Without this confirmation the RSS when a jammer is straight ahead would be the same as when a jammer is behind the antenna or even not present at all. Similarly to the research by Badawi, interference will be confirmed as present by using the Σ beam, which is very wide. In the case of the CTL3520 both elements are used for this confirmation, as opposed to the single element

used by Badawi. This simplifies the beamformer as the circuit does not need the ability to deactivate elements. Secondly, the output of the beamformer is the value of the RSS of Σ, minus the RSS of Δ. This creates a sharp peak broadside to the array and a relatively flat response elsewhere (as the magnitude of Σ and Δ are fairly similar except near to broadside). This peak creates an unambiguous indication of the direction of the signal.

The CTL3520 system uses this approach of using both Σ and Δ for direction indication, but is not capable of sweeping the beam. It is a handheld unit and the user manually sweeps the unit and determines the direction of the peak signal strength.

In terms of the exact requirements for the null beam, a narrower null will increase the resolution of the antenna by making the peak in the $\Sigma-\Delta$ output sharper. A narrower, deeper null will also increase the sensitivity of the antenna. This is because the total radiated power is relatively constant, but if less power is contained in the part of the beam pattern with the null, more of it is contained in the lobes of the beam. This pushes the flatter portion of the $\Sigma - \Delta$ pattern down by increasing the value of Δ, and so makes the peak more prominent.

To steer Σ and Δ away from broadside, a phase shift is applied to all the elements equally (with half of them also receiving the additional 180° shift to achieve the null). This steers the beam and null together. However, it becomes clear as the beam is steered further from broadside, that the main beam steered by a given amount from broadside is exactly the same as the null beam steered by that same amount from endfire. That is, it is not possible to apply a phase shift of more than 180° degrees because the Σ beam will look instead like the Δ beam. While mathematically this may not be a problem as the system will still be subtracting Σ from Δ, the problem is the ambiguity in the output. The Σ beam at this point would have one null broadside to the array, and two lobes of equal size pointing endfire. From this information alone, the system has no way of determining the AoA of the signal.

As a side note, it is actually possible to remove this ambiguity. By considering the phase of the incident signal, exactly which lobe it is in can be determined. However, this would require the signal to be digitised, something that should be avoided if at all possible as it dramatically increases the complexity and, more importantly, power consumption of the system.

The research in this Chapter will focus on using an electronically steerable phased array antenna, which produces both a Σ and a Δ beam, and uses the two of them together to increase the resolution of an otherwise physically small array.

2.3 Proposed antenna designs

For all of the designs proposed below, a simple patch element is used. An antenna element designed specifically for this work is discussed in Chapter 3. However, first it is necessary to verify that an array antenna will work as required. Hence in this Chapter a basic element is used for proof of concept, and it is assumed that the principles behind the array design (in terms of overall layout, if not exact dimensions) would not be significantly affected by the design of the patch. Based on these assumptions it is acceptable to design the array using this provisional patch design.

The basic patch antenna is the same as that used in the CTL3520 (with permission), shown

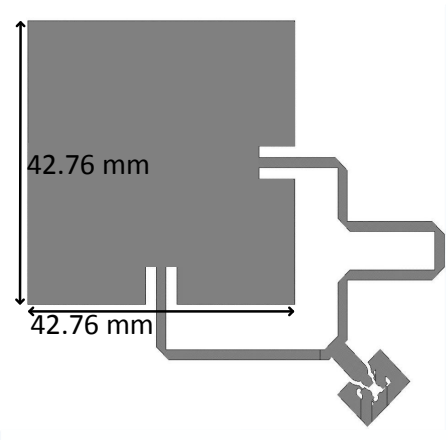


Figure 2.7: One element of the array. Each side is 42.76 mm, to make it resonant at GPS L1.

in Fig. 2.7. It is designed to be resonant at GPS L1. Both horizontal and vertical polarisations are present; they are combined at the surface mount u.FL connector to the bottom right of the image. The additional length in the horizontal path creates a 90° phase shift, resulting in a right-hand circularly polarised antenna. This polarisation is chosen because while jammers are usually linear, their orientation is unknown. By using a circular polarisation losses due to polarisation mismatch are minimised. The antenna is constructed on 1.6 mm FR4 with a ground plane on the reverse of the board. The manufacture of these designs uses a standard process, keeping monetary costs and manufacturing time to a minimum. This is important in keeping the costs of the system low as the development costs must be recouped eventually.

The system is designed to work at GPS L1. It has been found that all jammers work at L1, or are designed to (even if some have poor quality which reduces their efficacy [10]). If other GNSS are jammed it is in addition to GPS. Hence the prototype designed for this Thesis will work at L1, 1.57542 GHz. The principles behind the design will apply to all other GNSS frequencies with only small modifications required.

The array designs are simulated using NI AWR Microwave Office. The first two designs are planar and are simulated using AXIEM. However, the third design is not planar and so is beyond the scope of AXIEM. The Analyst 3D solver in AWR is used instead.

There are a number of key figures of merit that will be used to compare the designs in this Section. The first is the **null depth**. The depth of the null is given as the difference between the lowest point of the Δ beam, and the point of the Σ beam that lines up with that lowest point. This takes into account both the depth of the null and the gain of the Σ beam, and indicates the height of the peak of $\Sigma - \Delta$. The second important factor is the **width of the null**. This will be calculated as the width of the null at the point where the value is 3 dB up from the lowest point of the null. The third factor is the **width of the Σ beam**. The value of the Half Power Beam Width (HPBW) will be used, which is the point at which the gain is 3 dB lower than the peak. A wider beam (which therefore has nulls further apart) can scan over a wider angle and is preferable. The final feature which will only be discussed qualitatively is the **beamformer**. An estimate of the beamformer's complexity will be made for each design. This is important because a larger beamformer will consume more power and weigh more.

The designs and analysis presented in this Section were presented at the 12th European

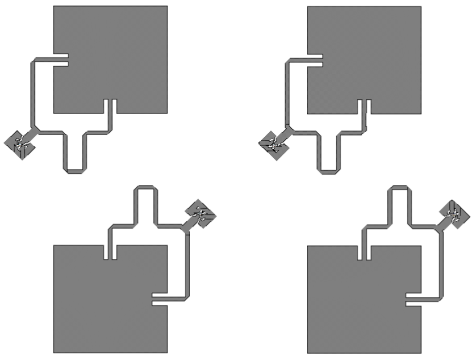


Figure 2.8: Four element array, square arrangement.

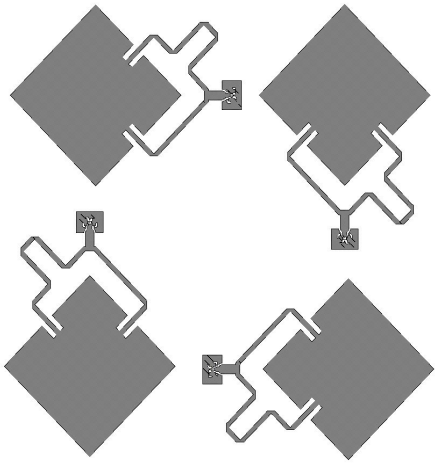


Figure 2.9: Four element array, diamond arrangement.

Conference on Antennas and Propagation in 2018 [43].

2.3.1 Design one - Four element Uniform Rectangular Array

The starting point for this design is the Chronos Technology CTL3520. This uses an array of two elements of the type shown in Fig. 2.7. The system uses the method of increasing the resolution described in Section 2.2.3.1. The CTL3520 locates the jammer by having the user scan the unit around, with the user determining the peak direction. However, with only two elements the system can only locate jammers in one dimension. By adding a second pair of antennas below the first pair, it should be possible to locate jammers in two dimensions. This is the simplest (and therefore smallest) two dimensional array for this purpose.

Two layouts were proposed. The first simply places a second pair of antennas below the first, as in Fig. 2.8. The second design rotates each patch through 45° (Fig. 2.9).

For the square layout, antennas could be used in pairs to search horizontally, then the pairs switched so that they could search vertically. For example, a horizontal search would be performed by combining signals from the two elements on the left, and with the two elements on the right also connected to each other. The null would be created vertically between the two pairs and could be steered. To search vertically the pairs would be swapped so that the top two are a pair, and so are the bottom two.

For the square layout, the inter-element spacing was $\lambda/2$, taking into account the dielectric constant, ϵ_r, of the substrate (approximately 4.47 for FR4). Fig. 2.10 shows the beam patterns for Σ and Δ for the square layout. The graph represents a slice through the beam, through the phase centre of the array in a plane normal to the array. It is broadly similar to the simulated version in Fig. 2.6. The sidelobes of Σ are small relative to the main beam so will not cause ambiguities. For this layout, the null depth is -48.8 dB relative to the top of the Σ beam, and the width is 0.31°. The Σ beam has a HPBW of 52.1°.

The diamond layout only uses two elements per dimension: it would perform two diagonal sweeps. One would use the top left and bottom right element, and the other would use the top right and bottom left. This could make the system faster as the beamformer could allow it to perform both horizontal and vertical searches simultaneously. It also increases the

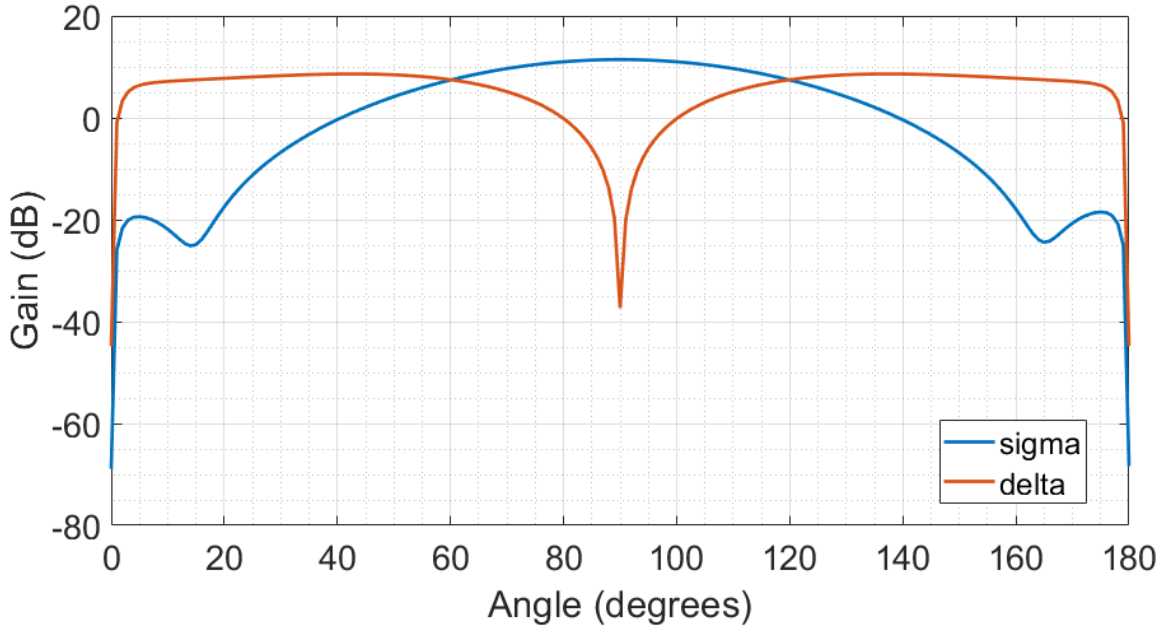


Figure 2.10: Σ and Δ beams for the square layout.

inter-element separation, which may improve the resolution. Fig. 2.11 shows the Σ and Δ beams for the diamond layout. The first thing to notice is that the increased inter-element separation narrows the main beam and increases the magnitude of the sidelobes as expected. The HPBW for this layout is 39.5°. The null is slightly less deep (-39.6 dB for the diamond, compared to -48.8 dB for the square layout) and slightly wider (0.544° for the diamond layout, 0.34° for the square).

Overall the square layout had better figures of merit than the diamond layout, so only this design will be carried forward to be compared with other designs.

The beamformer for this design is relatively simple. A block diagram to create all the necessary beams is shown in Fig. 2.12. The beamformer requires phase shifters to steer the beam. The centre crossover block is formed of switches and splitters to allow the pairs of antennas to be made either horizontally or vertically, then a set of combiners to allow the pairs of signals to be added together. The final block is a 180° hybrid coupler, which is discussed in more detail in Section 4.7. After the hybrid coupler, the signal is digitised. Amplifiers are not shown.

2.3.2 Design two - Seven element Uniform Circular Array

The second design is an experiment to see if increasing the number of elements improved the design, for instance increasing the gain (through summing more elements) and thus the sensitivity. The resolution may also be improved due to the wider aperture potentially narrowing the null, but without increasing grating lobes. The wider aperture may also suppress sidelobes.

Six elements are arranged in a hexagon with a seventh element in the centre. This antenna cannot do two equal, separate, horizontal and vertical sweeps but can do three sweeps separated by 60°, which would still allow a jammer to be localised in two dimensions. The antenna layout is shown in Fig. 2.14. A two dimensional slice of the beam pattern is

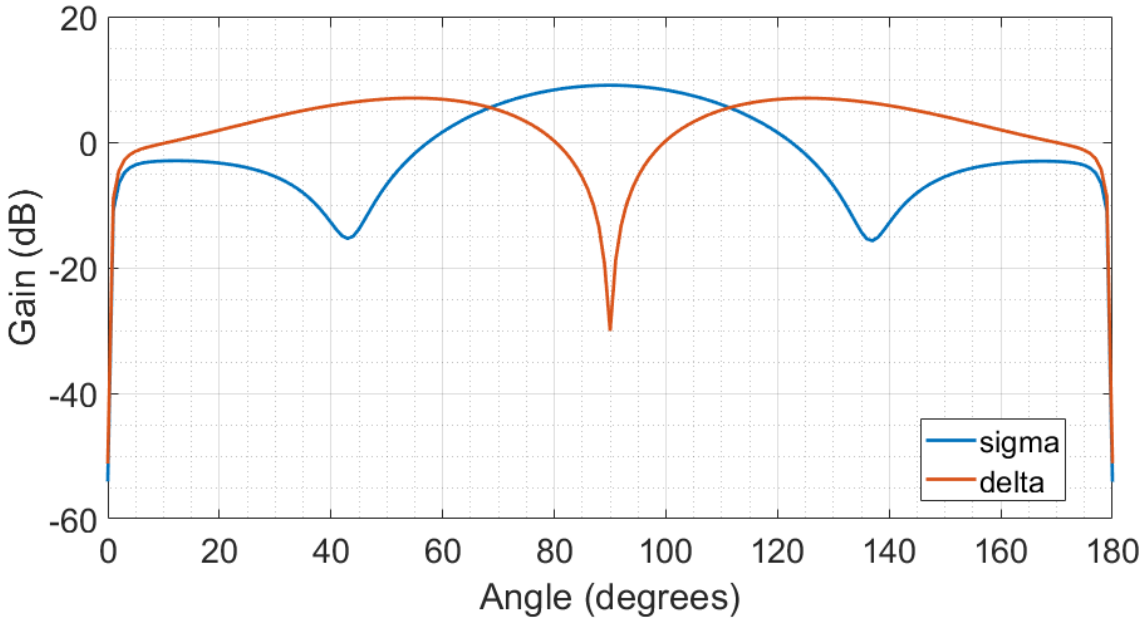


Figure 2.11: Σ and Δ beams for the diamond layout.

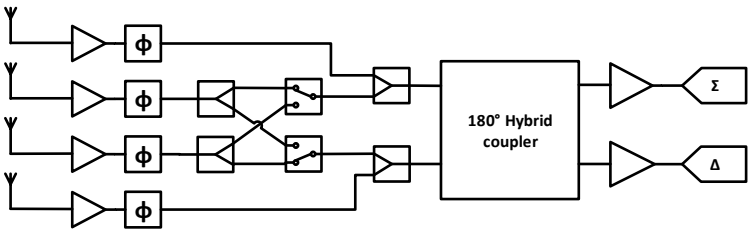


Figure 2.12: Minimum beamformer required to perform horizontal and vertical searches using the square layout.

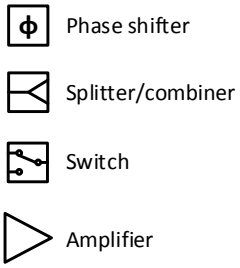


Figure 2.13: Key to symbols used in beamformer diagrams.

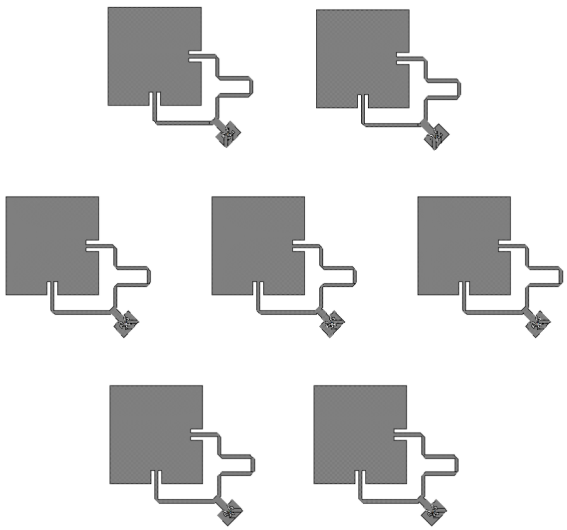


Figure 2.14: Layout of design two. Six elements are arranged in a hexagon with a seventh central element. All antennas have $\lambda/2$ separation to all their neighbours.

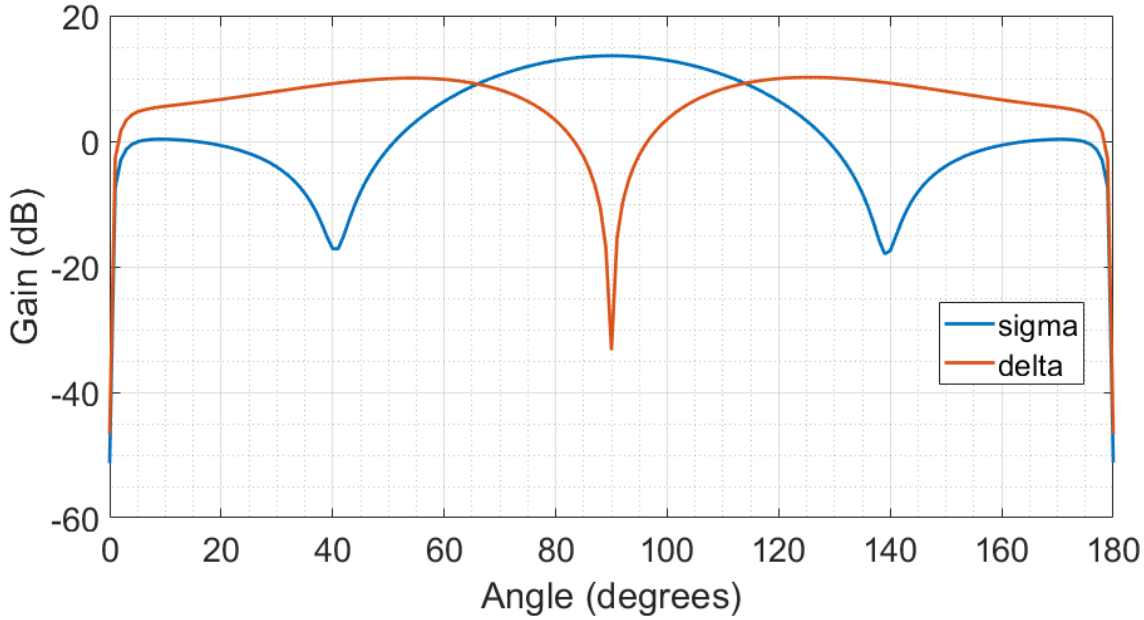


Figure 2.15: Σ and Δ beam for the seven element uniform circular array design.

shown in Fig. 2.15. The main beam, Σ, is created by adding together all the elements in phase. As expected the gain is higher with the higher number of elements (13.32 dB for seven elements, compared to 11.4 dB for the square layout). The null of the Δ is more complicated. If three elements are given a 180° phase shift relative to the other four elements, the null is not very deep (-17.0 dB) and is very wide (9.4°). To make a useful null the centre element must be deactivated while the RSS of Δ is being measured. With the centre element deactivated for the Δ beam the null depth becomes -48.28 dB and 0.323° wide. These figures are better than the four element array in design one.

The effect of steering this antenna is a weak point. When the null is broadside to the array it is in one plane, as with the four element design. Steering the null of the four element desgin has no effect on the shape of the null; it simply 'tips over' (Fig. 2.16). However, when the null of the seven element design is steered it becomes non-planar, as shown in Fig. 2.17. This 'twist' could be characterised but it would increase the computational effort required to determine the angle of arrival. A note regarding the three dimensional plots of beam patterns in this thesis: the colours represent the gain in a given direction, just as the distance between the phase centre of the array and the edge of the 3D view does. The scale for the colours has not been included because it varies between every plot and serves only to clutter the image unnecessarily. Hence the 3D views presented here are for demonstration purposes only.

Having more elements in the array increases the size of the array, making manufacture more expensive. It also increases the number of components in the beamformer. Fig. 2.18 shows an example beamformer. The elements forming the ring are numbered clockwise (or anticlockwise, and the starting point is unimportant) in order 1-6, with the centre element being number 7. With the square array it was assumed that two elements (1 and 4) would be fixed. For the seven element array, it is assumed that two elements of the ring that are directly opposite will be fixed. An expanded crossover section allows the fixed elements to be combined with any of the four closest elements either side.

An additional feature in this beamformer is a method of adding the centre element to

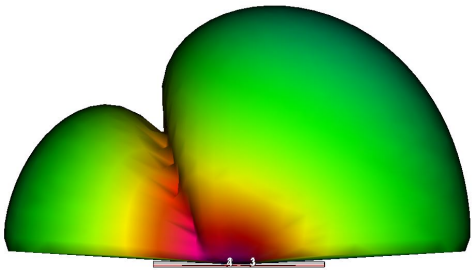


Figure 2.16: Effect of steering null of four element design. Colours are representative, not to scale.

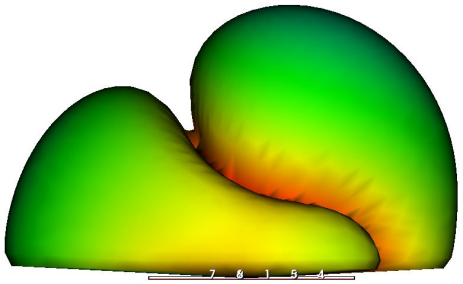


Figure 2.17: Effect of steering null of seven element design by the same amount.

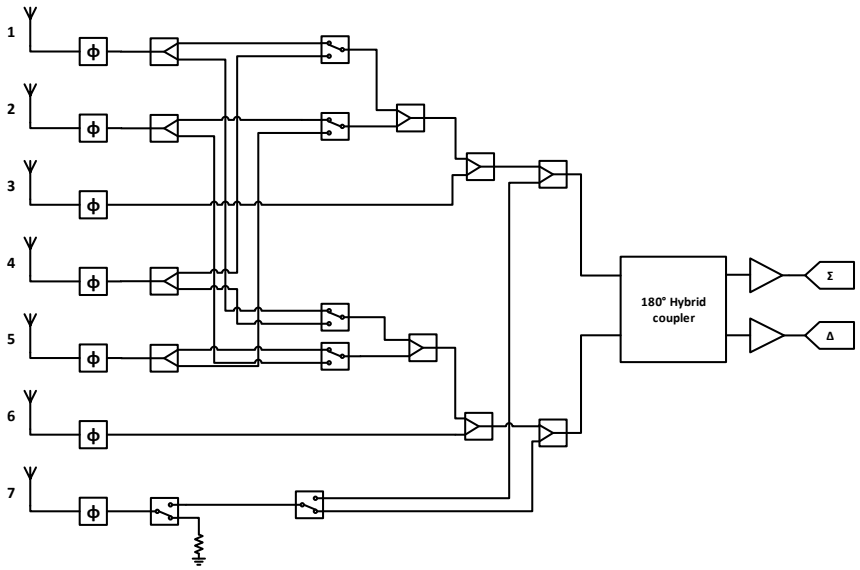


Figure 2.18: Possible beamformer for a seven element array.

one of the groups of three elements. However, this only applies when the value of Σ is being recorded. A switch can also take the output of the centre element and divert it to ground, which is necessary for the Δ beam.

A 180° hybrid coupler has the advantage that both Σ and Δ are produced simultaneously, reducing the time taken to read both. However, for this design the centre element must be deactivated for the reading of Δ, increasing the time taken. An alternative method would be to include additional phase shifters and splitters instead of the hybrid coupler but this would further increase the cost, complexity and power consumption of the beamformer.

Overall this antenna offers a slight improvement over the four element design in terms of some features of the beam pattern, but at a cost of increased size and power consumption, as well as slower operation.

2.3.3 Design three - Eight element faceted Uniform Circular Array

The third design takes a different approach. This PhD is researching a system that could be either mounted on a UAV to detect interference, or at a fixed location to mitigate it. This design would suit both. UAVs of the small, low cost, multirotor type do not fly very high - typically on the order of a few tens of metres. It has been mentioned that beam scanning angle is important to cover as large an area as possible, assuming the sensor is mounted facing

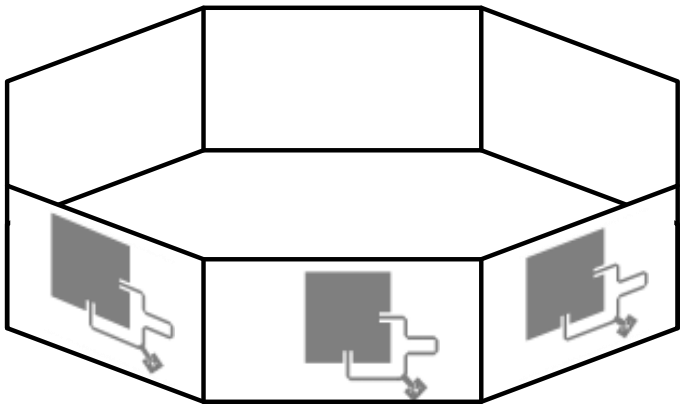


Figure 2.19: Diagram showing eight element faceted ring antenna array.

downwards. This limits the usability of the UAV as it would need to fly higher to increase the illuminated area. An alternative would be a system that scans in azimuth, rather than in x and y directions. A ring would also be applicable to mitigation as it would be able to detect jamming from any direction, which may occur in some environments.

This design was inspired by Sheleg *et al.* who proposed in 1950 a ring of dipole antennas that can quickly scan in azimuth [69]. Since this paper was written, technology has improved somewhat so their considerations for the beamformer can be disregarded, but the idea for the antenna is still applicable.

The third proposed antenna design uses a ring of patch antennas. The patches themselves are planar but there is a 135° angle between each element as shown in Fig. 2.19. Each 'face' of the octagon is formed of a 95×95 mm square of FR4 with a ground plane on the back and the antenna on the front. Curved arrays have been used before, for example by Byun *et al.* in [70]. Their proposal created a hemisphere with a number of patches placed on the surface. While the ideas presented in this paper are more applicable to a downwards-facing antenna, it implies that a curved array is capable of producing good (that is, narrow and deep) nulls and therefore mitigating jamming. The antenna designed in this Section is not curved but faceted, for ease of manufacture.

Instead of using all eight elements at one time, antennas would scan in adjacent pairs. Each element could be either the left or right half of a pair, meaning there is a total of eight pairs. This would limit the angle over which the beam has to scan to $360/8 = 45°$ per pair. This has the advantage that the beam does not have to be steered far from broadside: generally the beam distorts as it is steered away from broadside. The same hardware could be switched so that all eight elements use one beamformer. This would be small, light and low powered. Alternatively multiple beamformers could be employed to increase the speed of the search by exploiting the highly parallel nature of the design.

For the simulation, just two elements were modelled, as shown in Fig. 2.20. It was assumed that the remaining elements were sufficiently far away that they would not influence the beam pattern. This design was simulated using the 3D Analyst solver in AWR Microwave Office.

Inspecting a two-dimensional slice through the beam pattern gives a promising picture.

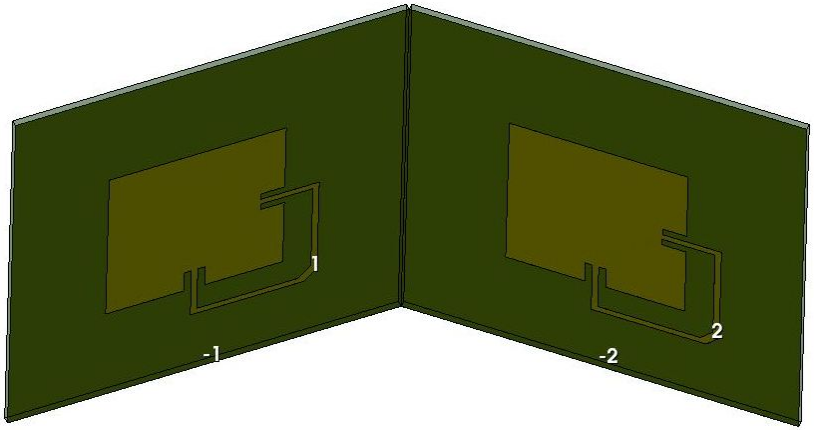


Figure 2.20: Three dimensional simulation used to determine the beam pattern for design three.

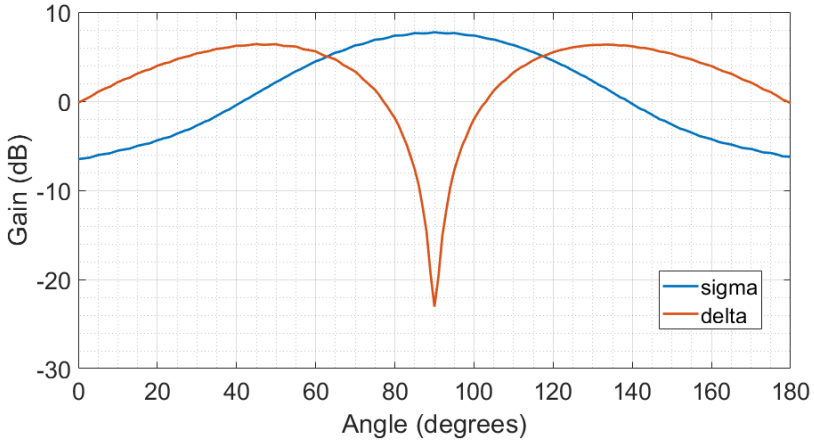


Figure 2.21: Σ and Δ beams for the eight element faceted ring design.

The null depth is -30.8 dB and it is 2.31° wide. The HPBW of the Σ beam is 57°, the widest of the three. This is because the peak gain of each element points in a slightly different direction, spreading out the power to a wider area. The null is wider and less deep than previous designs due to the antennas being 'tilted', which decreases the illuminated area and therefore effective aperture size.

While on the surface the Σ and Δ beam patterns appear to be similar to the previous designs, the null of the Δ beam is rather more interesting than the two dimensional slice would imply. According to the slice, the null depth is approximately -30.8 dB relative to the peak of the Σ beam. However, the true shape of the Δ beam, and thus the problem with this design, becomes apparent when the 3D simulation result is viewed. It is worth noting at this point that the results from Analyst consider the back lobes, unlike AXIEM. As AXIEM assumes an infinite conductive plane, the back lobes are not simulated, hence why this image differs from Figs. 2.16 and 2.17. The Analyst simulation (Figs. 2.22 and 2.23) shows that the null is only deep in the same plane as the array. This would make the antenna not very good at finding jammers if they are not in the same plane as the antenna. As the system could be mounted on a UAV that is flying above the area with the multipath, this antenna would not be effective at locating jamming.

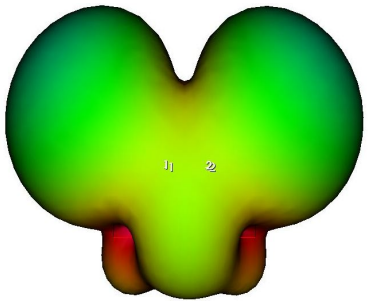


Figure 2.22: Side view of 3D beam pattern for the faceted eight element design. The antenna is pointing upwards, viewed from the top of the ring (cf. Fig. 2.19).

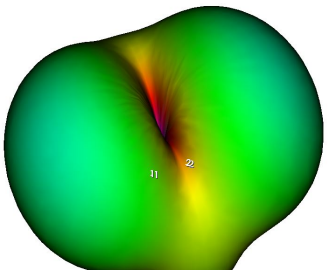


Figure 2.23: Oblique view of eight element array 3D beam pattern, showing null not present out of plane.

Table 2.1: Comparison of figures of merit for the different antenna designs.

	Design one	Design two	Design three
Null depth	-42.6 dB	-34.67 dB	-20.5 dB
Null width	0.31°	0.32°	2.31°
HPBW	52.1°	42.6°	57.5°
Beamformer	Moderately simple	Complex	Very simple

2.3.4 Comparison

Design three has the best HPBW, meaning it can be swept over a very wide area. However, this design sweeps in azimuth and each pair only covers 45°, meaning the sweep angle is irrelevant. It also has a very poor null - it is not very deep and it is wide compared to the other designs, meaning it will have poor resolution and sensitivity. In addition the null is only in the plane of the array, further decreasing its effectiveness. Therefore overall the good points of this antenna do not compensate for its drawbacks and it will not be considered further. Hence the decision is between designs one and two.

Both design one and design two have very similar null depths and widths, meaning the sensitivity and resolution will be similar. Design one has a slightly wider beam (as measured by the HPBW), meaning a larger area can be searched. However, the main advantage of design one is the complexity. Design two has no improvement in the beam over design one, and it comes at the cost of requiring a much more complex (and therefore expensive, heavy and power consuming) beamformer. It is also physically larger and so for two of the three design criteria (the physical size and weight, and the power consumption) it is significantly worse.

Hence, overall, the best design is the simplest: a planar, four element array of patch antennas. As this is the smallest possible design, there is no need to extend testing to smaller versions (unlike if the larger design had proved to be better, in which case the testing would have continued so as to find an optimum balance between beam pattern, and weight and power considerations).

2.3.5 Effect of coupling on beam pattern

It has been theorised that the amount of coupling between the elements of an array may affect the depth and width of a null. Hence this link will be investigated using these designs.

Table 2.2: Table evaluating the effect of coupling between elements and the null depth and width.

Antenna design	Coupling (dB)	Null depth (dB)	Null width (°)
Design one (horizontal pair)	-33.99	-40.24	0.279
Design one (vertical pair)	-29.16	-42.55	0.306
Design one (diamond type)	-31.90	-30.52	0.507
Design two	-29.18	-34.67	0.323
Design three	-50.42	-20.49	2.318

In Table 2.2, both variants of design one are considered. While the diamond type layout was not good for jammer detection, it still provides data for this comparison. For the square layout of design one, both horizontal and vertical pairs are considered. This was done because the orientation of the patches mean that some pairs couple more than others.

It was expected that increasing the coupling between elements would decrease the depth of the null. This would make sense because the null relies on subtracting the signal from one antenna from the other, but in the presence of coupling the two signals will not cancel out completely. As it is there was no correlation at all between the correlation and the null depth. It may be the case that the coupling has an effect, but the effect is smaller than that caused by the array geometry. Hence the coupling is affecting the null depth, but the changes in geometry between the different arrays completely masks any trends. Further investigation could be carried out in future work.

2.4 Monopulse method

Up to this point it has been assumed that the search will be carried out by sweeping a planar null and establishing the direction of arrival as the result of multiple sweeps. There is another method: monopulse radar. The name is based on the fact a single snapshot can establish the location of a target. By combining the signals from all four elements in different ways, the location of the jammer can be found. The beamformer for a monopulse system is significantly more complex, requiring multiple 180° hybrid couplers (which are discussed in more detail in Section 4.7) [71]. Aside from the additional complexity, a beam created from all four antennas has much poorer results compared to the versions presented above. Using the layout named Design one, the previous method produced a beam with a null that is -40.24 dB deep, compared to -25.4 dB for a monopulse design using the same array. The null width shows a difference that is even more dramatic: the previous method produces a null that is approximately 0.3° wide, but with the monopulse layout the null is 14° wide, which is not useful for this work. As a result monopulse radar will not be discussed further in this Thesis.

2.5 Conclusions

In this Chapter a number of different designs for arrays of patch antennas were tested. They were simulated and compared based on figures of merit such as the beam pattern they produce, and what hardware would be required to control them. It was found that a small,

simple design of four elements arranged had very low size, weight and power requirements without compromising on the beam pattern and therefore this design has been chosen.

Chapter 3

Design of an antenna element for jammer detection and classification

In which: Requirements are discussed . . . Polarisation measurements are described . . . Antenna fabrication methods and limitations are considered . . . A design is manufactured and measured . . . Results are presented

In Chapter 2 an array layout was designed using a basic patch. According to simulation the design is suitable for direction of arrival estimation, which would be done by sweeping a beam and noting the angle with the highest incident power. While simulations indicate that this array could be effective, it is physically large. The aim of this work is to design a system that could be mounted on a UAV to allow for rapid searching of highly scattering environments; a physically large antenna will not be ideal for this purpose. The antenna is also not suitable for anything other than direction of arrival estimation.

The searching method proposed in Chapter 2 requires no digitisation of the signal and only uses the power. In this Chapter, the literature is searched to see what other information can be gathered about an incident signal without requiring any digitisation. It is found that the polarisation of the signal can be calculated using just power measurements, and so this avenue is pursued. The polarisation may help with identification of jammers, but this would require additional experimentation with real jammers to prove.

It had been determined previously that for this situation a patch antenna was the optimum antenna type due to its low profile, simple manufacture and low cost. A number of different techniques for designing a patch, each with their own benefits and drawbacks, are studied in this Chapter. It was found that one design (a slot coupled patch) was objectively the best but required a larger amount of computational resource during the design stage than was available for this project. Instead the second best design (a probe fed patch) was designed and manufactured. A high-dielectric substrate provided by Premix was used so that the antenna could be made smaller and lighter.

The design was simulated and found to work as needed. An antenna based on the simulated design was manufactured using the Premix substrate and tested. It was found to operate at a significantly different frequency to what the simulation indicated. Extensive debugging identified an air gap as the potential source of the problem but the time required for a second iteration, as well as the difficulty in testing the working antenna fully, meant the

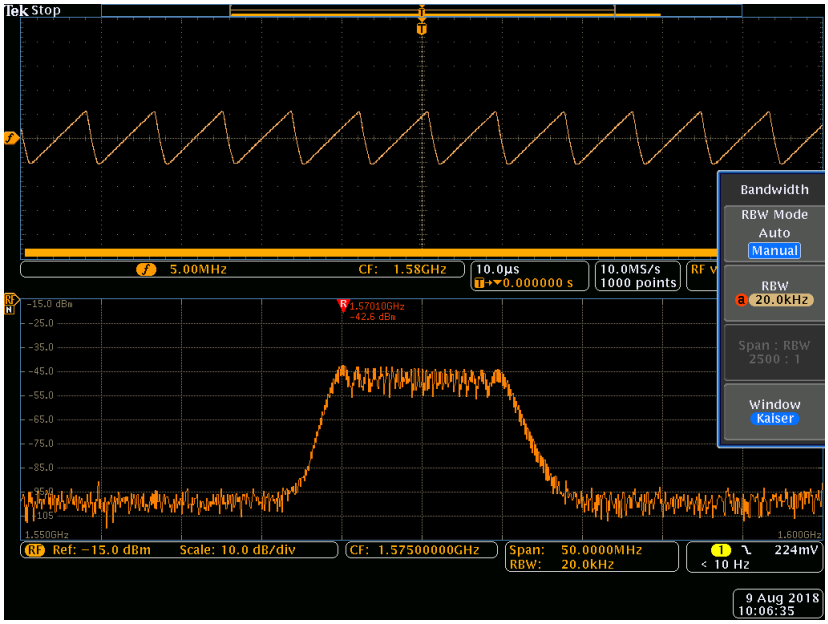


Figure 3.1: Chirp and frequency spectrum for a commercial jammer.

design was not pursued further.

3.1 Background

The aim of this research project was to design a system that can detect and mitigate an interference source. Mitigation can take multiple forms: null-steering could block out the source of interference. Alternatively the source of the interference could be localised and removed. Finding the source would be made easier if more information about the type of signal was available. For instance, testing may show that a re-radiating antenna (a known source of interference [11]) has a different signal to a true jammer. Tests of many commercially available jammers showed that they had chirped signals [10]. An example jammer output, which was captured during trials at the Sennybridge Training Area in Wales, is shown in Fig. 3.1. Jammers also typically had dipole type antennas, which would produce a linearly polarised signal. Spoofed signals may also be linearly polarised, which could help distinguish them from legitimate GPS signals, which are right-hand circularly polarised. It would also be useful to detect certain characteristics about the signal for localisation purposes. For instance, if the same signature was measured at multiple locations, the progress of a jammer across the country could be tracked and the vehicle identified. Hence there are two possible pieces of information that could be gleaned from the signal: the polarisation, and the chirp parameters. A chirp detector could be made using any antenna, instead depending on the beamforming hardware. In contrast, measurement of the polarisation of the signal requires an antenna capable of particular polarisions. This Chapter will focus on the polarisation measurements as a new antenna design is required for miniaturisation purposes, and polarisation measurements have certain requirements for the antenna design. Therefore it is important to consider polarisation at this point.

There is interest in detecting spoofing. Spoofing is the act of producing false GPS signals with the aim of fooling receivers into giving an incorrect time or location. While it is currently

rare and has only been known to be carried out by nation states, the rise of low-cost Software Defined Radios (SDRs) makes spoofing accessible to a wider range of people [72].

Multiple methods of detecting spoofing have been proposed. Different solutions use different portions of a GNSS receiver to detect the spoofing signal. For example, the front end of most GNSS receivers will record the C/N_0 (Carrier to Noise) ratio, which is an indicator of the received signal quality. The C/N_0 will vary over time as atmospheric conditions and satellite positions change. Sudden changes in the C/N_0 will indicate that something is amiss [73].

More sophisticated detection methods consider the results after the receiver calculates the time and position. Some receivers will incorporate error detection systems such as Receiver Autonomous Integrity Monitoring (RAIM), which detects malfunctioning satellites. RAIM could be used to detect spoofing, but assumes only one or two satellites will be producing incorrect signals and so will likely be unable to detect spoofing [28].

If a receiver is fitted with an Inertial Measurement Unit (IMU), relative motion of the system can be measured. This can be compared with the motion according to the GNSS receiver [74].

Still other methods rely on detecting anomalies within the result without external hardware for comparison. For instance, Cavaleri *et al.* look for distortions in the early/late detection signals [75]. A GNSS receiver will produce three replicas of a signal with different phases relative to the last known phase of the received signal. One will be slightly early, one slightly late and one on time. The strongest correlation will indicate if the phase delay between transmitter and receiver is increasing, decreasing or static. The function produced during this process is the Cross Ambiguity Function (CAF). During the spoofing process the attacking transmitter will attempt to duplicate the true signal. When the phase delay is similar enough for the receiver to be fooled, the CAF will become distorted. This distortion can be detected and used to identify a spoofing attack.

Another system uses a different aspect of the decoded GNSS signal to detect spoofing. Receivers calculate a very accurate time from the GNSS signals and spoofers may wish to change that (or may accidentally distort it - an SDR may use a less accurate clock than the atomic clocks carried by GPS satellites). If a system, such as a wide area power network monitoring system, had multiple distributed receivers, they could compare their calculated times to identify any discrepancies [32].

A final method is to determine the angle of arrival of the GPS signals. The true GPS signals will come from a number of different directions, corresponding to the different satellites. Unless a highly sophisticated attack is made, all of the signals from a spoofer will come from the same direction. Using a method such as MUSIC (as discussed in Chapter 2) would allow the receiver to determine if signals are coming from satellites or land-based receivers [76]. If the signals were determined to come from one direction, this information could also be used for mitigation.

All of these spoofing methods require information that is unavailable in the proposed system. Using the CAF, the received time, cross-correlation with an IMU, or the C/N_0 would require the system to decode the GPS signal. The current system only uses the power in the signal to keep it as simple as possible. The size, cost and power consumption of the

system would increase hugely if any of these approaches were used.

A method that has not yet been proposed is the polarisation. It is not guaranteed to be useful in all situations, but it may be that spoofers tend to use a linearly polarised antenna, meaning a malicious signal could be detected and mitigated. This hypothesis, that the polarisation of the signal could be used to identify the source of interference, will be tested in this Chapter.

So what is polarisation?

3.1.1 Measuring polarisation

Consider a point-to-point line of sight link, with a transmitter T and a receiver R. The power received, P_r can be expressed as (3.1),

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi r)^2} \tag{3.1}$$

Where P_t is the transmitted power, G_t and G_r are the gain of the transmitter and receiver respectively, λ is the wavelength of the transmitted signal and r is the distance between the transmitter and receiver. The equation states that the received power is a function of what is transmitted ($P_t G_t$), plus the gain of the receiver (G_t), plus the power loss between the two points ($\lambda^2/(4\pi r)^2$). However, this equation assumes that all of the signal power that arrives at the receiving antenna is absorbed; that is, that the antennas are matched. However, this is not the case, but it can be accounted for mathematically by adding a mismatch term, m. The value of m is $0 \leq m \leq 1$, where a value of one indicates the antennas are perfectly matched. If they are less than perfectly matched the value of m will be lower.

This mismatch may be as the result of polarisation. Polarisation is the property of a wave that describes the orientation of the oscillations. The polarisation of an antenna describes the sort of waves that can be received by the antenna [77].

To calculate the mismatch, the power of the incident signal needs to be redefined. The incident power flux density (S_{rec}) is now given by (3.2).

$$S_{\text{rec}} = \frac{P_t G_t}{4\pi r}[\mathbf{I}] \tag{3.2}$$

$\mathbf{I}$ is the normalised Stokes vector. It comprises four values, called the Stokes parameters, (3.3).

$$[\mathbf{I}] = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \tag{3.3}$$

The Stokes parameters can also be represented on a Poincaré sphere, Fig. 3.2.

Using the Poincaré sphere, the values of the Stokes parameters can be calculated as in (3.4).

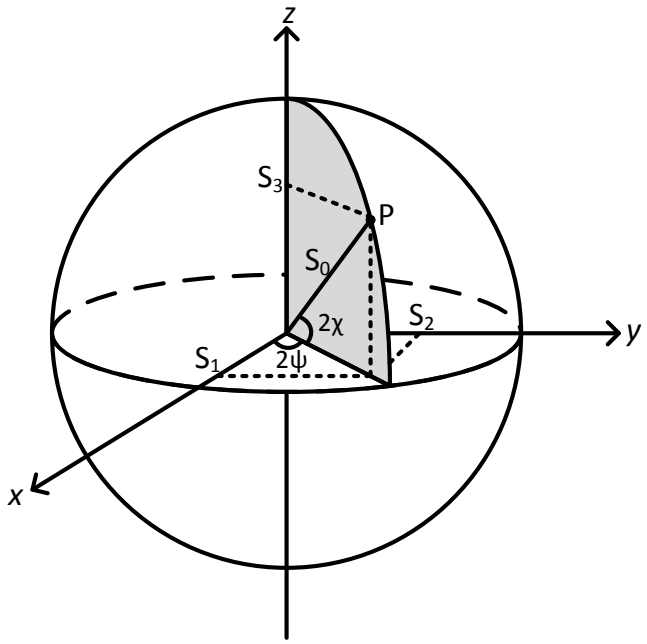


Figure 3.2: Poincaré sphere representation of polarisation.

$$\begin{aligned} S_0^2 &= S_1^2 + S_2^2 + S_3^3 \\ S_1 &= S_0 \cos(2\chi)\cos(2\phi) \\ S_2 &= S_0 \cos(2\chi)\sin(2\phi) \\ S_3 &= S_0 \sin(2\chi) \end{aligned} \tag{3.4}$$

The four parameters together describe the polarisation of the signal. The total intensity of the signal is S_0. It is often normalised to one. S_1 and S_2 describe the horizontal and vertical polarisation respectively, while S_3 gives the circular polarisation.

The polarisation of a signal can be a combination of these values. A signal can also not be completely polarised, which would be indicated by a value of S_0 that is less than one. The Stokes vector for the true GPS signal would be $\mathbf{I} = [1\ 0\ 0\ 1]^T$, indicating that all of the power is in the right-hand circular polarisation. If the polarisation is a mixed type, the values of the parameters S_0 to S_3 will be between 0 and 1.

There are two ways to measure the polarisation: using coherent or using incoherent measurements. The 'coherency' of the measurements in this case refers to the phase. For coherent measurements the amplitude and phase of the horizonal and vertical polarisations needs to be measured. This can be translated into a point on the Poincaré sphere, and therefore a Stokes Vector. However, this requires measurement of the phase, and this project is attempting to minimise cost and power consumption by not measuring the phase of the signals.

The second method of measuring polarisation is incoherent with respect to phase - only the power in the received signal is measured. To find all the values, a number of measurements are made to find the polarisation [78]. The power in horizontal, vertical, diagonal (+45°), and two circular polarisations are measured. This requires an antenna with the horizontal and vertical ports separated, and a beamforming network capable of producing all of these polarisations.

For the horizontal and vertical polarisations, one connection needs to be excluded from the measurement. For the diagonal and circular polarisations, the horizontal and vertical signals must be added together with a phase shift of 0°, 180°, 90° or −90° applied to the vertical polarisation for +45°, −45°, RHCP and LHCP respectively.

3.1.2 Possible antenna designs

The next stage in this project is to design an antenna that meets the requirements of the system (small, light, and low cost), but is also capable of polarisation measurements. Incoherent measurements will be used. In this Section, a number of antenna types will be considered and one will be chosen.

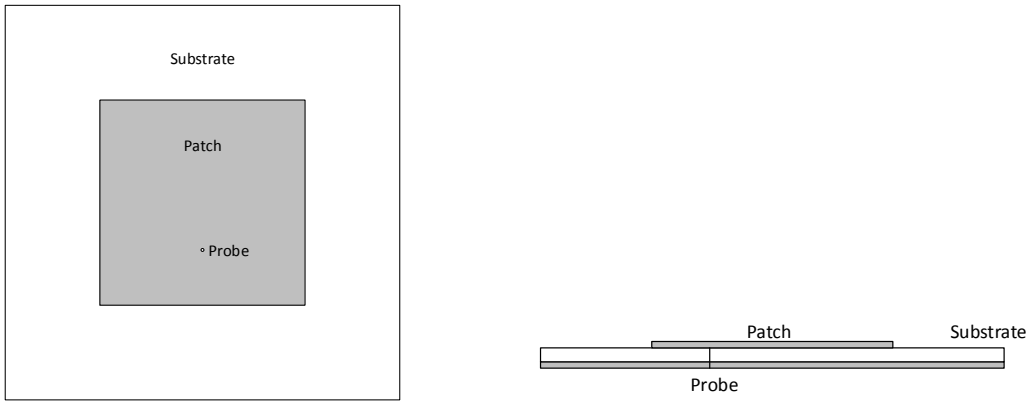
An array can comprise any sort of antenna. For instance the White Alice Communications System comprised 180 antennas, each an array of waveguide slot antennas [79]. Another famous system, the Duga (or 'Woodpecker') over-the-horizon radars, were a set of dipole arrays. Both of these radars are very large and were made a long time ago (1958 for the White Alice Communications System, 1976 for the Duga radar). They were also very high power - 50 kW and 10 MW respectively. As a result the design requirements were different and modern technology was unavailable. For this system a microstrip patch antenna will be the best option, due to their size, weight, cost and flexibility [80].

A patch antenna is a planar antenna manufactured on a circuit board, with a ground plane on the other side of the board. This board could be FR4, the standard material, or it could be a more unusual substrate. The ability to manufacture antennas using standard PCB processes significantly reduces the cost. It also has the advantage that it is all automated, making the results more reliable than hand-made dipoles or other alternatives. A third benefit of patches is their ability to be incorporated into unusual structures. For example, it is possible to manufacture arrays on non-flat surfaces. Abbaspour *et al.* created a patch antenna array on a cylindrical surface [81]. Two patches were placed on top of one another, increasing the bandwidth. This sort of stackup is possible for patch antennas due to their flat shape, and has the ability to significantly increase the bandwidth. Patch antennas typically have a narrow bandwidth but in this work a bandwidth of 0.57 - 2.6 GHz was achieved.

The patch antenna created by Abbaspour was probe fed, but there are other feed arrangements possible. The antenna used in Chapter 2 was edge-fed, using an inset to match the impedance of the feed structure. A third design, slot-coupled antennas, uses a slot in the ground plane to couple signals from a microstrip trace to a patch.

3.1.2.1 Edge fed antenna

An edge fed antenna is the simplest design to manufacture. The design process, however, requires some care. The point at which the feed joins the antenna should be inset into the antenna to have the maximum power transfer and minimum reflections. This is done by impedance matching - the centre of the antenna is effectively zero impedance while at the edge the impedance is at a maximum. At some point in between the impedance will be 50 Ω, and this distance must be found, either through calculation or simulation. However, this inset increases the effective length of the outline of the antenna, changing the resonant frequency.



(a) (b)

Figure 3.3: Top (a) and side (b) views of a probe fed antenna.

It will also affect the current distribution as the cuts cause disturbances, which could reduce the polarisation purity [80].

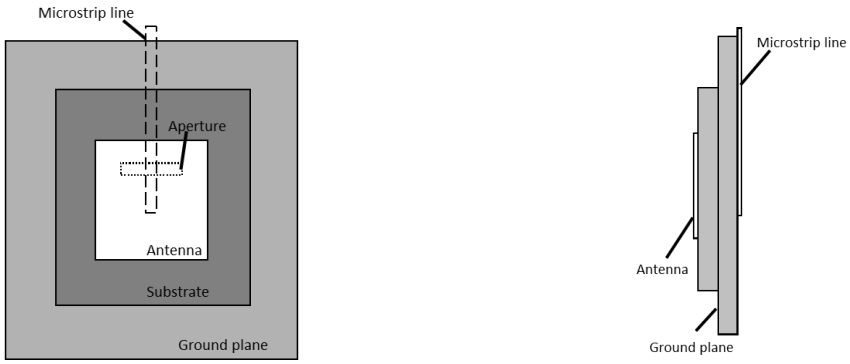
This type of antenna has a tradeoff that cannot be avoided. The best patch antennas are fabricated on thicker substrates with lower dielectric constant values, where the ground plane is electrically further from the antenna. This increases the radiation efficiency. However, the microstrip feed circuit will have fewer losses (that is, it radiates less) if the substrate is thinner. As such one part of the system, either the feed or the antenna, will not be optimal.

3.1.2.2 Probe-fed antenna

To avoid the problem of the increased edge circumference when using an edge-fed antenna, the system can instead be fed with a probe, as in Fig. 3.3. The probe can be in the form of a via, which is simple to fabricate. The feed network would then be on the back of the antenna (with a ground plane between, perhaps made using a multilayer PCB). A probe feed has a smaller effect on the radiation pattern of the antenna than an edge-fed antenna, although it will still cause slight aberrations around the feed point [82]. As with an edge-fed antenna, the impedance of the antenna can be matched to the feed network by making the probe the correct distance from the edge. The location of the probe will also affect the polarisation of the antenna. This design has the advantage that as the antenna and the feed network are on different layers, both can have their substrate thicknesses optimised.

3.1.2.3 Slot coupled antenna

A third way to feed a patch is to use a slot coupled antenna. Slot coupled antennas were first proposed by Pozar [83]. This is a modified version of a slot antenna. A slot antenna has a cut in the ground plane. This interruption of the current return path of the feed network causes it to radiate. A slot coupled patch antenna places a patch above this ground plane slot. This design normally has better polarisation purity than other patch designs, at the cost of increased fabrication costs. It also has advantages over a standard slot antenna: a slot antenna radiates both left and right with respect to the antenna in Fig. 3.4b. By coupling the signal into a patch above the ground plane, more of the signal is made to radiate left (outwards from the patch) increasing the gain of the antenna.



(a) (b)

Figure 3.4: Top (a) and side (b) views of a slot coupled antenna, adapted from [83].

A slot coupled antenna is more complicated to design than an edge-fed antenna because there are more factors to consider. First of all, the patch: in this case the patch must be square so that it resonates at the correct frequency in both horizontal and vertical polarisations. This is the case for all of the designs. However, the interactions between the patch and the other components increase the complexity. The slot should be centred under the patch to maximise the coupling. However, it is not possible to place both slots underneath the centre as they cannot overlap. The length and width of the slot also determine the amount of coupling [84] so it must be designed to maximise the coupling, within the previously mentioned constraints.

The fabrication of this antenna is also more complicated than an edge-fed antenna due to the increased number of layers. As shown in Fig 3.4b the antenna is on the top layer, with a substrate separating it from the ground plane. An additional layer of dielectric substrate separates the ground plane from the microstrip feed. The main advantage of this design is that it allows the antenna to be fabricated on a thick substrate and the feed network on a thin substrate, optimising both parts of the system. It also has the advantage that as there are no discontinuities in the patch, meaning the radiation pattern is closer to isotropic (which is ideal in this case).

3.2 Antenna designs

Having reviewed the possible methods and their benefits and drawbacks, the next stage is to design the antenna. Typically the size of a patch is dictated by the wavelength of the signal; the length of the element is normally between 1/3 and 1/2 of the wavelength of the signal [80]. Changing the dielectric will therefore change the effective wavelength and therefore the overall size of the patch required. The system requires the antenna to be as small as possible and so a good dielectric is preferable.

3.2.1 Materials

By choosing a more suitable substrate, the antenna can be made physically smaller without affecting its electrical size and therefore its performance. Whether a substrate is 'good' or not depends on its dielectric constant, ϵ_r, also known as the relative permittivity. The dielectric constant of free space is ϵ_0 and they are related as in (3.5) and (3.6), where ϵ is the absolute permittivity of the material.

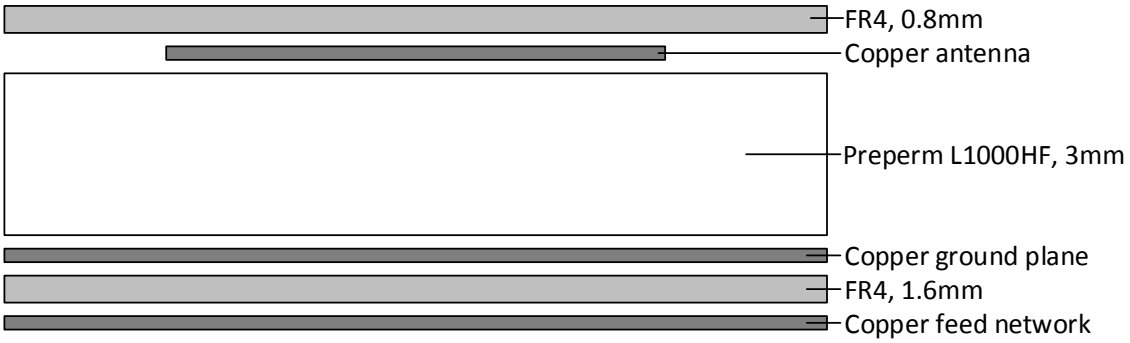


Figure 3.5: Stackup for antennas designed in this Section. Thicknesses not to scale.

$$\epsilon_r = \frac{\epsilon}{\epsilon_0} \tag{3.5}$$

which rearranges to:

$$\epsilon = \epsilon_0 \epsilon_r \tag{3.6}$$

The speed of a wave propagating in a medium is c_r, as calculated in (3.7), where μ is the magnetic permeability of the medium.

$$c_r = \frac{1}{\sqrt{\epsilon\mu}} \tag{3.7}$$

Hence as the dielectric constant increases, c decreases. A decrease in the propagation speed translates to a decrease in the wavelength:

$$\lambda = \frac{c}{f} \tag{3.8}$$

Hence if the dielectric constant ϵ_r is increased, the wavelength will decrease and the antenna can be made smaller. With this in mind a sample of Preperm L1000HF [85] was acquired from Premix. This high performance substrate has a dielectric constant ϵ_r of 10, much higher than the ϵ_r of approximately 4.47 of FR4. The dielectric constant of this material is similar to that of high performance ceramics, but it has a number of advantages in this particular application. It is approximately 40% lighter than an equivalent ceramic. It is also not as brittle, meaning it would be more durable if the UAV platform had a hard landing.

Preperm L1000HF is a plastic substrate. The equipment required to bond copper directly to plastic was not available. A many-layered stackup that allowed the use of standard FR4 was designed instead. The antenna is fabricated on a piece of single sided FR4. The copper side is placed directly against the Premix so that the maximum effect of the dielectric is achieved. On the other side of the Premix substrate, a double sided piece of FR4 is used. The top side (placed against the Premix substrate) is the ground plane for both the antenna and the feed network, and the opposite side of the FR4 is used for the feed network. This stackup is drawn in Fig. 3.5.

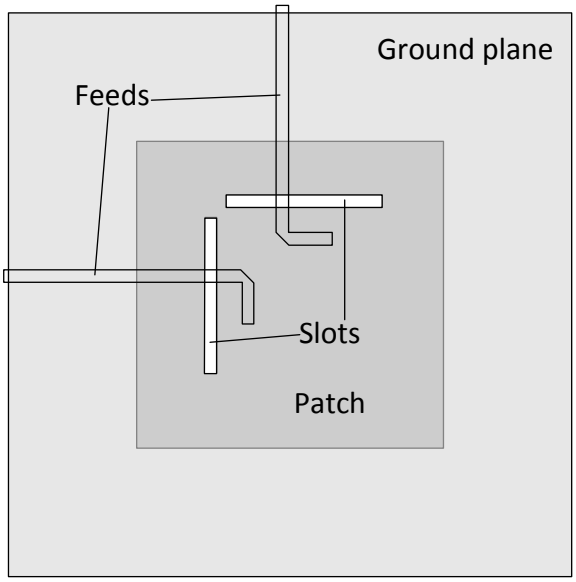


Figure 3.6: Proposed dual polarisation aperture coupled patch antenna design.

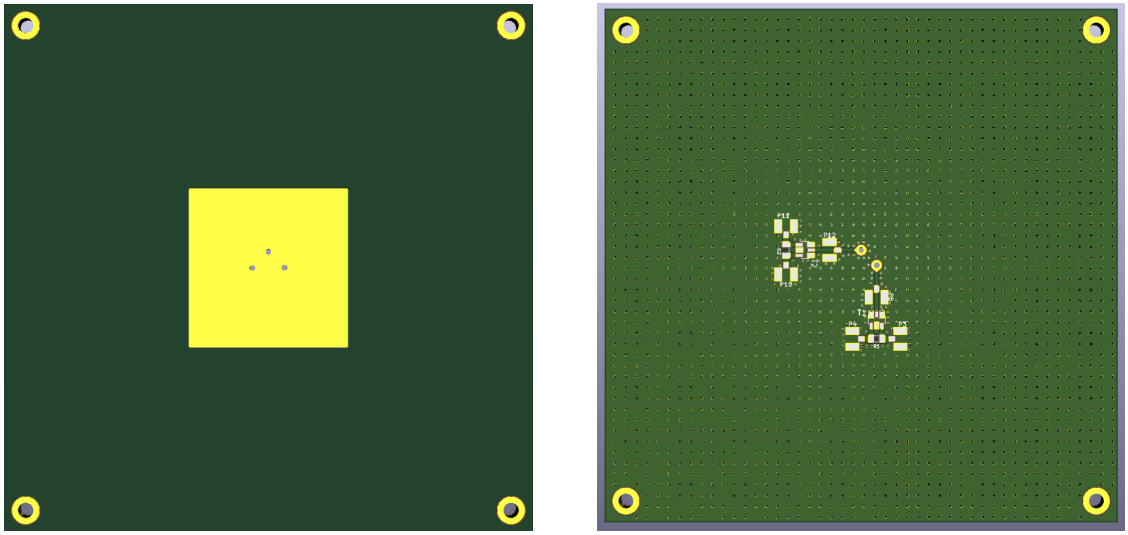
3.2.2 Designing the antenna

An attempt at designing a slot coupled antenna was made. The first pass made an antenna with just one port and therefore one linear polarisation. The performance of this antenna was passable, with a return loss of -14.2 dB. However, the second version required the addition of a second port. The design is shown in Fig. 3.6. The ideal slot size meant the two slots would overlap. Moving the slots sideways reduces the coupling between the feed and the patch. To achieve the maximum amount of radiation across the slot, the slot must be a certain distance from the end of the trace (depending on if the trace has a short or open termination). To achieve this length the trace must bend. The bend causes a discontinuity in the trace which increases radiation, some of which may couple into the other feed. These two factors make this antenna difficult to tune. It was decided that this antenna was not worth the time to perfect as it would require significant tuning to achieve good performance, and each simulation took a significant amount of time due to its complexity. Hence at this point the aperture coupled patch design was shelved.

As the aperture coupled design was not possible within the constraints of the project, the second best solution, a probe fed antenna, was designed. The antenna was simulated in AWR Microwave Office. Once the parameters for the design had been finalised, it was drawn in KiCAD. The feed network was also designed in KiCAD. The two parts of the design, the antenna and the feed network, are shown in Fig. 3.7. Both boards are square, and 95 mm on each side. They also have four 3 mm holes for mounting.

Fig. 3.7a shows the antenna is a square, 28.5 mm on each side. The holes are positioned at (11.4, 14.2) and (14.2, 11.4) relative to the top left corner of the antenna. The third (rightmost) hole is in the same position relative to the top right corner and exists to avoid any orientation errors when having the PCB fabricated and the subsequent assembly of the whole design.

Fig. 3.7b shows the feed network. This is a two layer PCB; the back is one continuous ground plane. The front (visible) side is also mostly covered by a ground plane. The two



(a) (b)

Figure 3.7: The antenna (a) and feed network (b) for one dual-polarised probe-fed antenna.

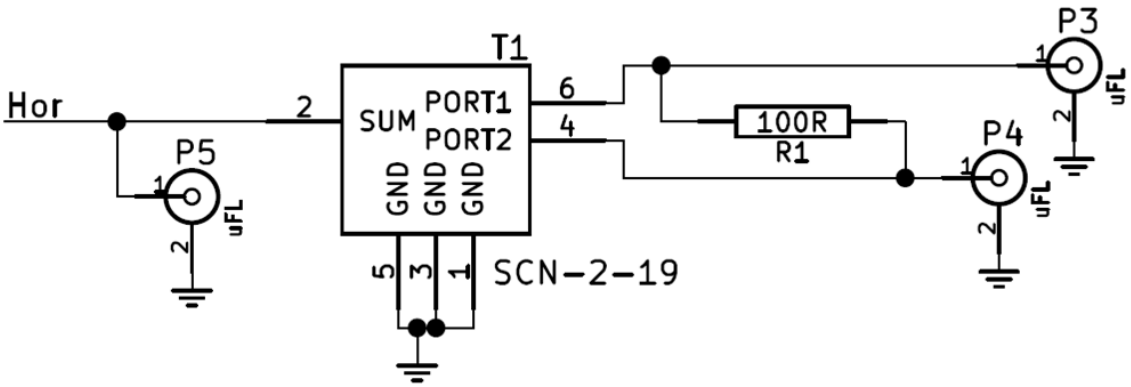


Figure 3.8: Schematic for one channel of the antenna feed.

planes are stitched together with vias to ensure there are no ground loops or other adverse effects. The circuit for the horizontal feed is shown in Fig. 3.8 (the vertical feed layout is identical). The feed is designed so that there are two choices: The first option is to populate just connector P5. This allows all of the signal to be taken off to the beamformer. Alternatively P5 is not populated and instead P3 and P4 are used. With this option T1 is also populated. T1 is a Minicircuits SCN-2-19+ 0° power splitter. It is a miniaturised Wilkinson coupler and requires one external matching resistor (R1). Hence if this option is used the signal from the antenna is split in half and can be carried to the beamformer via P3 and P4. For the measurements in this Chapter only P5 was populated. While there is a small stub that passes under P5 and to pin 2 of T1, it was assumed that as the length was short it would not cause any significant loss of performance.

3.3 Antenna performance

The antenna was tested for three aspects: the return loss at the GPS L1 frequency, the polarisation purity, and the ability of it to function as part of an array and produce the required beam patterns.

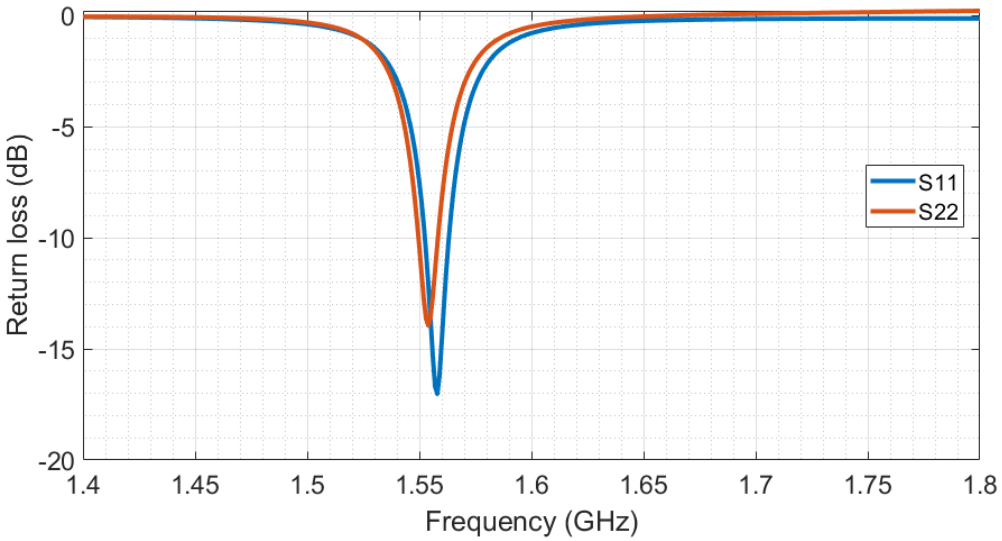


Figure 3.9: Return loss of both ports of the probe fed patch antenna design based on an AXIEM simulation.

3.3.1 Simulations

Fig. 3.9 shows the return loss for both ports of the probe fed design. The peak return loss is roughly centred on GPS L1, although port 2 has a worse match, and at a slightly lower frequency, than port 1. This could be tuned out in later iterations of the design. The peak return loss is -17.0 dB for port 1 and -14.0 dB for port 2. For comparison, the return loss of the patch used in Chapter 2 has a return loss of -18.6 dB. However, this figure also includes any losses in the feed network as the feed has not been de-embedded. Hence the two antenna designs have similar return losses. However, the bandwidth of the old design is wider, at 12.9 MHz compared to the new design's 5.2 MHz. An attempt was made to increase the bandwidth using a stacked patch arrangement, but similarly to the aperture coupled design, the simulations proved too time consuming and so it was shelved until it proved necessary.

The axial ratio refers to how much (or little) cross-polarisation exists. For example, to achieve a good axial ratio, there should be almost no vertical polarisation when just the horizontal mode is excited. An axial ratio of one (*i.e.* 0 dB) indicates that there is pure circular polarisation. Fig. 3.10 shows the axial ratio of the antenna. At the lowest point it is 3.4 dB, and it remains within 3 dB of this value over 100.4°. This indicates that if the jammer is significantly far away from broadside, the polarisation measurements will have errors introduced by the system and for accurate measurements, the antenna should be pointed at the signal of interest.

The coupling of the two ports was also measured. If the coupling is too high, even with a good axial ratio the measurements will still be inaccurate. The coupling between ports is shown in Fig. 3.11. Both S21 and S12 are plotted but according to the simulation they are identical. A real antenna will have slight imperfections that will cause them to differ. The coupling at the frequency of interest is low (at a minimum of -25.29 dB at 1.57°). This should be sufficient for the polarisation measurements despite the possible manufacturing variations.

The final test assessed how well the antenna functions as part of an array. For this test four patches were placed in a square (the arrangement that was found to be the optimum

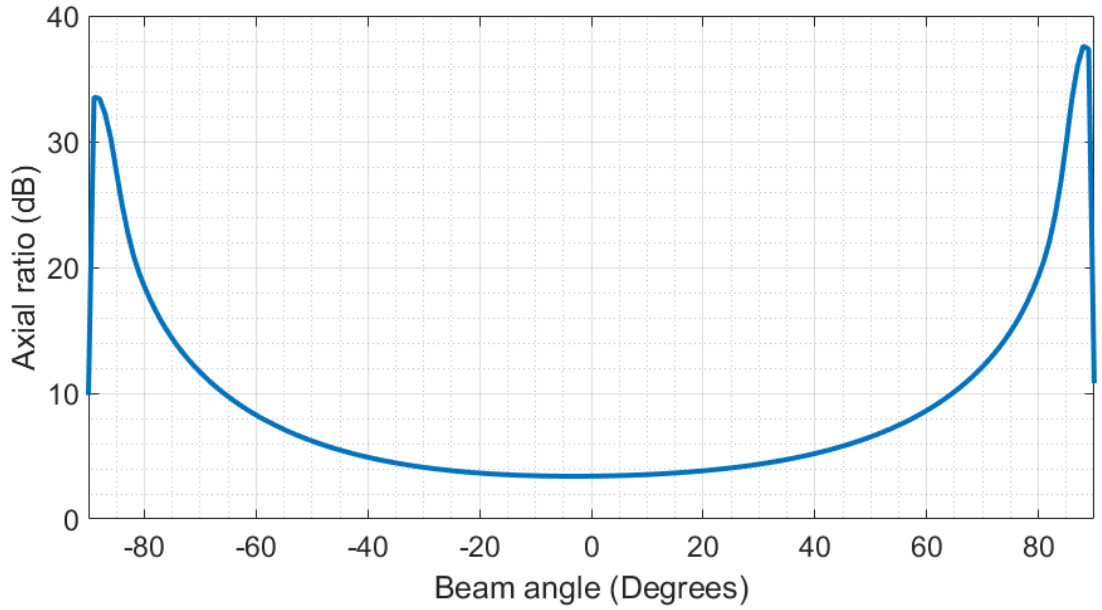


Figure 3.10: Axial ratio for probe fed patch antenna design.

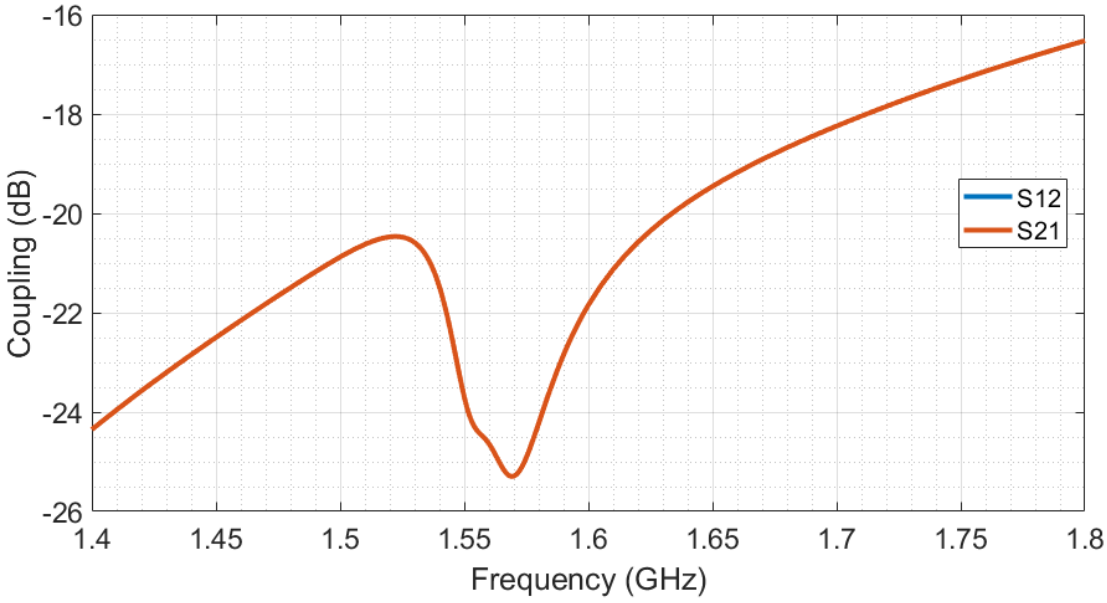


Figure 3.11: Coupling between ports 1 and 2 of the probe fed patch antenna design.

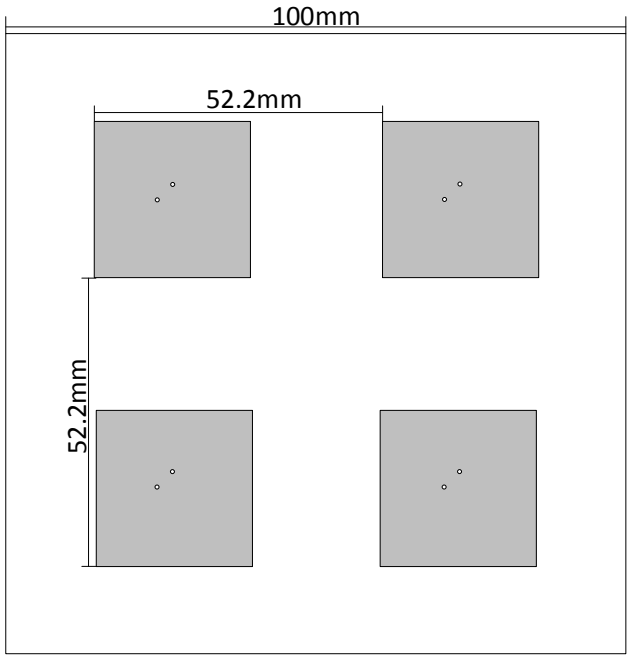


Figure 3.12: Arrangement of four flexible-polarisation patches in an array for direction finding.

Table 3.1: Figures of merit for Δ beam patterns with different polarisations.

Polarisation	Null depth (dB)	Null width (°)
Vertical only	-14.16	4.13
Horizontal only	-12.25	6.80
Circular (matching)	-12.81	5.06
Circular (opposing)	-	-

geometry in Chapter 2). The design is shown in Fig. 3.12. The antennas were then excited in a variety of different ways to reproduce how the antenna might be used in a real situation. This means both Σ and Δ beam patterns were drawn, with their parameters assessed.

It was found that the polarisation purity can be maintained when a Σ beam is produced. If all of the antennas produce a circular polarisation, the axial ratio is 0.93 dB broadside to the array. The lowest value was 0.26 dB at -19°. It remained within 3 dB of the axial ratio at broadside over 104°.

In terms of beam pattern, the peak gain of the Σ beam was 7.76 dB in both circular and linear polarisations. In contrast, the Δ pattern varied depending on the type of polarisation used. The results of different polarisations are summarised in Table 3.1. The beam pattern was measured so that the plane of the null was vertical, and the 'slice' taken through the radiation pattern was therefore horizontal. In this Table, two circular polarisations were tested. The first ('matching') had all the array elements excited with the same handedness of polarisation (in this case, left hand circular polarisation). In the second test, the left two elements were left hand circularly polarised, while the right two elements were right hand circularly polarised.

When only the vertical ports were excited the null depth and width were the best, at -14.16 dB and 4.13° respectively. Other polarisations produced worse results, with one circular polarisation producing no discernible null at all.

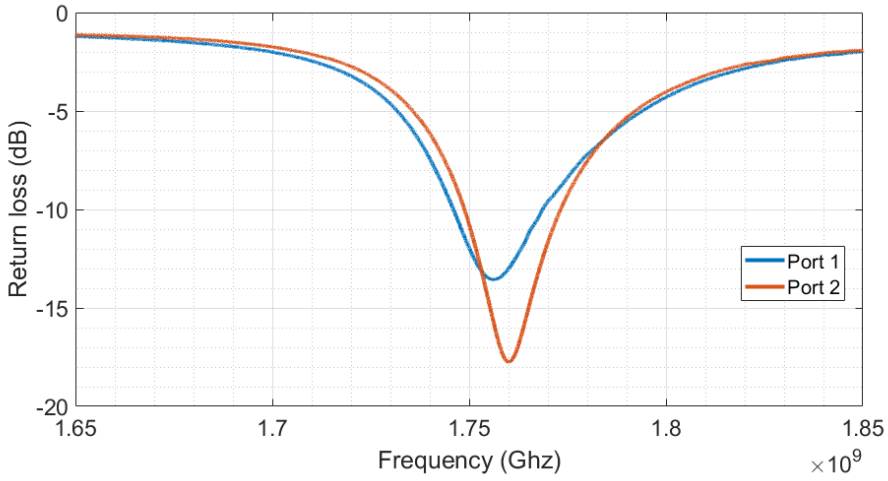


Figure 3.13: Measured S11 of one polarisation of one element of the miniaturised antenna.

The inter-element coupling was also measured. According to the simulation the polarisation did not change the amount of coupling, which was measured between two horizontal ports of adjacent antennas. The coupling was measured as between -24.78 dB and -27.42 dB in the 20 MHz band centred on GPS L1. For comparison, the same layout in Chapter 2 had interelement coupling of -34 dB. The increased coupling is most likely due to the fact the elements are positioned closer together. The null depth (measured from the bottom of Δ to the peak of Σ) was -40.24 dB for the patch design used previously, compared to -21.92 dB for the new array layout. This lends some evidence to the theory that increased coupling between elements decreases the null depth.

While this antenna is capable of producing the beam patterns required for direction of arrival measurements, they are affected by the polarisation of the antenna. Hence polarisation measurements would be carried out using just one element, but the whole array would be used for direction finding.

Having measured the antenna's performance in simulation, the next stage was to manufacture it and verify that it meets the requirements.

3.3.2 Testing of hardware

The antenna was fabricated as described above, with one single antenna in the centre of a 10 cm piece of Premix substrate. The probe feeds were created using wires soldered through the whole stackup. The return loss was the first parameter to be measured. It was measured by connecting one antenna port at a time to a Copper Mountain Planar 304/1 Vector Network Analyser (VNA).

The results of the measurements (given in Fig. 3.13) showed that the antenna radiated not at L1 but instead at approximately 1.77 GHz. This is around 200 MHz away from the target frequency of 1.575 GHz. At this point work moved from testing the antenna to diagnosing the problem. A number of different strategies were applied and their results are described here.

Resoldering the antenna. The first assumption made was that there was some fault in the process of building the antenna. An air gap between the antenna and the premix sub-

strate would reduce the effective dielectric constant, making the antenna electrically smaller. As a smaller antenna radiates at a higher frequency, this was considered to be a reasonable assumption. To remedy this the wires holding the antenna to the ground plane were desoldered. The entire stackup was held with clamps to ensure there were no gaps, then it was resoldered. The resonant frequency of the antenna decreased to 1.74 GHz.

Resizing the antenna. The next modification changed the electrical size of the antenna not by changing the dielectric, but by changing the physical size of the antenna. The patch was confirmed to be the same size as the simulated design so that was not the source of the error. The amount the antenna should need to be increased by was calculated based on the current size and frequency. Copper tape with conductive adhesive was placed around the edge of the antenna to increase the size. Solder was added to ensure conductivity. The antenna was then reconnected with the feed network, using clamps as before. While this experiment did decrease the resonant frequency to approximately 1.67 GHz, this was less than the calculated effect. This may have been due to the copper tape and solder increasing the thickness of the antenna and therefore introducing an air gap even when clamps were used.

Verifying materials. The manufacturers of the substrate, Premix, were contacted. It was suggested that the material may be incorrect. It was weighed and found to be the correct product, so it can be assumed that the dielectric matched the value used in simulation. However, this assumption cannot be tested as it was not possible to measure the dielectric of any substrate - neither the Premix or the FR4 could be confirmed to have the correct specifications.

Modifying the simulation. It seemed clear that there was nothing obviously, fundamentally wrong with the construction of the antenna. The next assumption to be tested was that the simulation matched the fabricated design. It was verified that the fabricated antenna was the same size as the simulated antenna, and that all material thicknesses were the same. Once this was confirmed, the next step was to modify the simulation to ensure it was identical to reality.

The initial simulations had been carried out using the AXIEM simulator within AWR Microwave Office. AXIEM assumes that all planes are infinite, including all of the dielectric layers. A second simulation was created using the Analyst simulator, which uses a more accurate drawing and includes the extents of substrates. This new simulation produced the same result as AXIEM, indicating that the infinite planes assumption is not the cause of the mismatch.

It was noted that the manufactured antenna had screws to hold the stackup together, which had not been entered into the original simulation. The screws were approximately 40 mm from the edge of the patch, positioning them in the near field of the antenna and therefore giving the potential for coupling to affect the resonant frequency. Screws were added to the AXIEM simulation. The resonant frequency according to that simulation was 1.57 GHz, meaning the screws were not the problem.

It was observed earlier that there may be air gaps between the layers that would change the resonant frequency. An air layer was added to the simulation, between the antenna and the Premix high dielectric substrate. The thickness of it was varied to find out what air

gap would be required to detune it by the required amount. Simulations showed that a gap of 0.1 mm (approximately the thickness of a sheet of paper) produced a resonant frequency of 1.78 GHz, which is similar to the results obtained with the manufactured antenna. It is possible that a gap this large was present even with the clamps if there was some surface roughness. There is other evidence to support this theory: adding an air gap increased the 3 dB bandwidth of the antenna from 5.2 MHz to 14.5 MHz. Figure 3.13 shows that the bandwidth of Port 1 was 10 MHz, while Port 2 had a bandwidth of 20 MHz. Finally, the thickness of the assembled antenna was slightly greater towards the centre: it was measured as 5.579 mm at the edge of the antenna and 5.587 mm halfway between the edge and the centre. (It was not possible to measure the very centre of the antenna.) Overall the weight of evidence suggests that there is some air gap present, even when the antenna is soldered together using clamps.

Unfortunately the time spent attempting to diagnose the problem with this design meant that there was not sufficient time to test it. It was especially difficult as the equipment available at the University of Bath would not produce good results and so arrangements would have to be made to make the measurements at another facility.

3.4 Conclusions

Having designed an array in Chapter 2, this Chapter aimed to design a smaller antenna element so that the system could be as small and lightweight as possible. It has been suggested that measuring the polarisation of an incident jamming signal could provide some information about the emitter in question. Hence when designing the new antenna element, it was made so that polarisation measurements would be possible. A number of different types of patch antenna were studied and it was decided that a probe-fed patch antenna would be made. While simulations were promising, measurements of the actual antenna did not match the simulations. The cause of the mismatch was found to be caused most likely by a small air gap between the antenna and the substrate. However, due to the time taken to diagnose the problem, iterate the design to solve the problem, and then test it (which required facilities not available at the University of Bath), the decision was made to move on from this part of the work.

For the rest of this research, the basic patch antenna used in Chapter 2 will be used.

Had more time been available, there would have been two desirable tests. The first would be to measure the polarisation and beam pattern of the antenna that was produced, without any modifications. The polarisation measurements would be taken at the frequency at which the antenna was resonant (around 1.77 GHz). The second test would be to fabricate a new antenna that is physically larger, so that it is resonant at the desired frequency (1.575 GHz). Again, the beam pattern and polarisation would be measured. In addition, the polarisation of different GPS jammers could be measured to determine if the system is capable of distinguishing them.

Chapter 4

Design of a reconfigurable RF front end

In which: block diagrams for several arrays are designed . . . Key elements are identified . . . Suitable hardware is chosen and designed . . . Hardware is tested individually . . . The hardware is combined with the antennas . . . Jammers are found

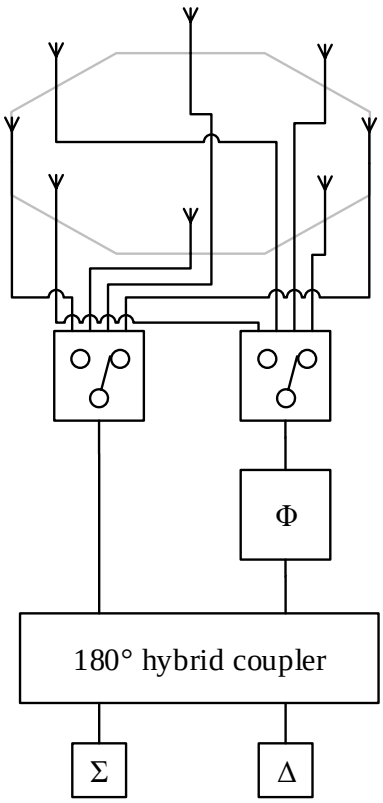
This Chapter describes the design of a set of components that would allow testing of the array and antennas designed in Chapters 2 and 3. While the designs have been simulated, it is important to compare their real-world performance to prove that they are suitable for the task. Simulations have indicated that one array design is preferable, but it may be that in reality performance is so poor that it needs to be redesigned. Additionally, this is first and foremost a research project and so if new ideas present themselves during the course of the work it is important to be able to test them with the minimum of additional effort. As a result, a modular approach to the front end design was used.

Example beamformers had been proposed during the array design process. Some of the components used were common across the designs, such as phase shifters, switches, and signal splitters. Examination of the literature showed that these components are used in many RF systems. Therefore it was decided that these components would be the first to be designed and fabricated. These components would also enable testing of the polarisation-measuring capabilities of the antenna designed in Chapter 3.

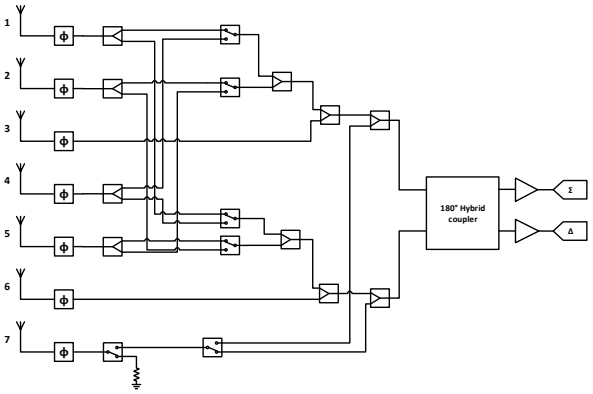
A variety of methods for building RF beamformers are available, such as waveguide and microstrip. Taking into account the low frequency and low power of the signals being used, as well as the requirement for a low cost system, it was decided that the best approach would be coplanar waveguide manufactured on FR4 substrate. The individual components were implemented using COTS devices which can be easily mounted onto circuit boards, keeping manufacturing time and cost to a minimum.

A set of boards were designed. Each contained a number of identical components, and connectors. This design allowed beamformers with different configurations to be designed and created quickly by simply switching the order of components. Each board was tested and the performance compared against the theoretical performance given in the datasheet to ensure the design of the system did not impair its operation.

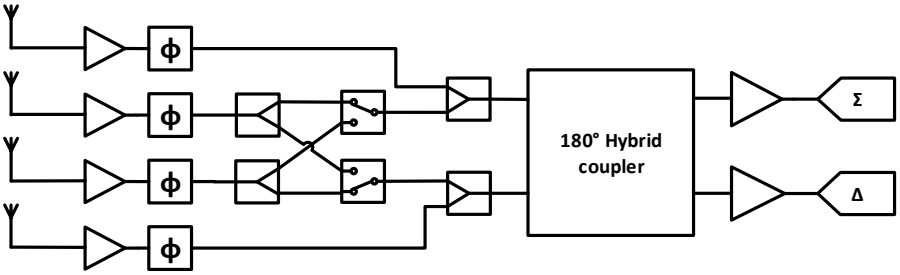
Having tested the design and found it met the requirements, a complete system was as-



(a) Beamformer for the eight element faceted ring design.



(b) Beamformer for the seven element circular array design.



(c) Beamformer for the four element square array design.

Figure 4.1: Three designs for beamformers from this work.

sembled. The array designed in Chapter 2 (made using basic patches and not the polarisation measurement-capable patches discussed in Chapter 3) was simplified and connected to the modular beamformer. Using code implemented on a PC, a simple sweep was performed. It demonstrated the system's ability to detect and localise the source of interference, using a real jammer in a realistic scenario.

4.1 Existing designs

The first stage of designing a beamformer is to identify the required components. The beamformers proposed in Chapter 2 are shown again in Fig. 4.1. They comprised phase shifters, switches and power splitter/combiners. However, the literature must be reviewed to ensure that no other additional components are required.

Sheleg proposed a ring antenna [69] that was the basis of one of the array antenna designs suggested in Chapter 2. While the array itself proved to have poor performance for this

work, the beamformer is still of interest. The proposal was to excite current modes in all the array elements that, combined, would result in one peak, using the principle of Fourier series'. These modes were generated using a set of variable phase shifters to control the signal fed (via a Butler matrix) to each element of the array. This paper did create hardware to test the design but it was very large (however, dimensions are not given in the paper). The phase shifters were also slow to operate.

Additional sources state the same required components for beamformers. For both planar and conformal arrays, phase shifters are required to produce the correct beam pattern [86]. In addition, conformal arrays may require amplitude and polarisation control. This source also mentions the introduction of MMIC (Monolithic Microwave Integrated Circuit) technologies, meaning electronic control of beams is now significantly faster and cheaper than in the 1960s, when Sheleg was producing his antenna. New wafer chemistries such as Silicon-on-Insulator are offering continuous improvement over more traditional RF (Radio Frequency) IC (Integrated Circuit) technologies such as Gallium Arsenide [87]. This will be the type of components used in this Chapter. High dielectric circuit board substrates such as Rogers compounds further reduce losses from coplanar waveguide or microstrip line [88]. This allows common PCB manufacturing techniques to be used to higher frequencies than previously possible. However, for the initial design standard FR4 substrate will be used.

It appears that phase shifters are required. However, this is not always the case. In [89], Vu demonstrated that nulls can be steered without using phase shifters, instead using amplitude shaping only. This would reduce the efficiency of the array overall by attenuating some elements. It also halves the number of nulls that can be controlled and requires them to be in conjugate pairs. The final, and largest problem with applying this work here, is it is not intended to steer the main beam, making it unsuitable. This work was carried out to overcome low resolution phase shifters, which is not expected to be a problem now, 35 years after the paper's publishing date in 1984. Hence this method will not be discussed further and instead only steering using phase shifters will be considered.

4.2 Requirements for building blocks

The device must be as low cost as is feasible. This is a low cost system, so the cost of the beamformer must be kept as low as possible without compromising on the system's performance. This includes the noise performance of the devices - a low cost device may introduce too much noise and make measurements unreliable.

The system must be small and lightweight. As with the antenna element and array design, it is important that the system is small and lightweight. This will make it more suitable for mounting on a UAV.

It must be possible to control the system rapidly. To be able to effectively locate and mitigate interference, the system must be capable of quickly finding the angle of arrival. If the delay in finding the jammer is too great the system being protected may lose lock. In a GNSS-denied environment, the signal to noise ratio (SNR) must be much higher to gain lock than to maintain it [10], and for the mitigation to be effective it must prevent a receiver from losing lock.

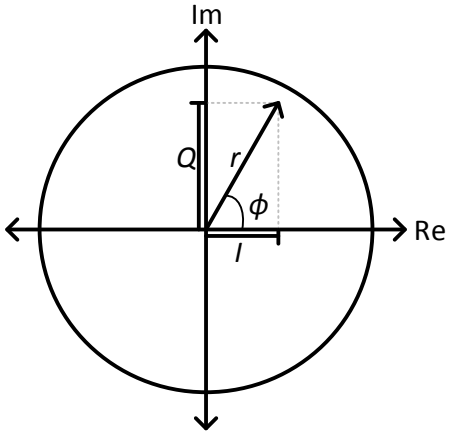


Figure 4.2: Demonstration of how I and Q inputs can modulate the phase and amplitude of an incoming signal.

4.3 Phase shifter

The phase shifter is required to steer the beam. There are two main groups of high frequency phase shifters available: discrete phase shifters, and vector modulators. Both will be considered here. A good phase shifter will be able to change position quickly, as well as being low cost and low power. It should have good accuracy and repeatability, so that the phase shift produced is close to the expected value and does not drift with time, or change each time a particular value is selected.

4.3.1 Analog Devices AD8341

The AD8341 [90] is a vector modulator that can operate between 1.5 and 2.4 GHz, covering the L1, G1, E1 and B1 bands. It comprises both a phase shifter and attenuator in one package. Two inputs, referred to as the In-Phase and Quadrature (I and Q) signals, control the exact modulation of the signal. The values of I and Q can be plotted on an Argand diagram, as in Fig. 4.2. The angle of the resultant vector, ϕ, gives the phase shift. The magnitude of the vector, r, gives the attenuation, where a larger r indicates a smaller amount of attenuation. As a result the modulation of the signal can be very high resolution, restricted only by the signals input to I and Q. The device has a settling time of 45 ns, giving a maximum switching frequency of 22 MHz.

The power consumption of the AD8341 is 0.625 W according to the datasheet. This is high as each element of the array will need to be connected to a phase shifter, meaning the smallest design would require 2.5 W for the phase shifters alone. However, the fact this is a vector modulator and can also control attenuation means the overall component count may be smaller, which will reduce the power consumption of the rest of the beamformer.

The minimum attenuation is -4.5 dB. That is, when the magnitude of the vector, r, is one, the insertion loss of the device is 4.5 dB. This would result in a loss of sensitivity of the system compared to phase shifters with a lower insertion loss.

For this component, I and Q are differential baseband analogue signals. Thus each phase shifter requires two differential Digital-to-Analogue Converters (DACs), which can be controlled electronically. These additional components increase the cost, size and power

consumption of the system. The RF inputs also require external matching components in the form of a 1.2 nH inductor on each line. However, the cost of passive components such as capacitors, inductors and resistors can be considered negligible.

The phase shifters will also require power supply components. Switch mode power supplies are more efficient than linear regulators. However, the switching that is vital to their operation injects noise into the output voltage. To avoid this, linear regulators will be used despite being slightly less efficient. A device such as the MCP1755 [91] is capable of supplying 300 mA at 5V, which is sufficient. A separate power supply IC would be required for each vector modulator, but this would aid in reducing coupling between channels.

4.3.2 Peregrine Semiconductor PE44820

The Peregrine Semiconductor PE44820 [92] is a phase shifter rather than a vector modulator. It uses a series of set phase shifts that can be activated or deactivated based on the digital input to the IC. It can only change the phase in steps of 1.4°, as opposed to the nearly continuous phase shifts available to the AD8341. It also cannot control the attenuation. The PE44820 is designed to work between 1.7 and 2.2 GHz but the output can be corrected using lookup tables, making it suitable for use down to 1.5 GHz, meaning it is also able to cover the main frequency bands of all the existing constellations.

The PE44820 is a lower cost device than the AD8341, which is reflected in its lower flexibility. However, in many ways it outperforms the vector modulator. The PE44820 has a power consumption of just 720μA, which is almost $1/1000^{\text{th}}$ that of the vector modulator. It also requires no external components as the phase shift is digitally controlled. The insertion loss is less than 5.5 dB at L1, which is slightly higher than that of the AD8341 and so would result in a lower sensitivity. It has a settling time of 365 ns, but the serial control restricts the maximum switching frequency to 25 kHz.

The lower power consumption of this device compared to the AD8341 means that a linear regulator could supply several phase shifters. However, in the interest of reducing coupling of RF signals through the power supply, separate regulators would be used for each channel. In this respect there is no price or size difference between the two options. Many of the performance metrics of this device can be improved with the addition of a negative supply rail, in particular the power consumption and switching frequency. Hence if it is found that performance needs to be improved, it can be at the cost of some more components.

The 1.4° minimum step size is a limitation. However, a 1.4° phase shift applied to half of the elements of the array results in a shift in beam angle of 0.5° (based on simulation of the antenna design chosen in Chapter 2). This level of accuracy may be sufficient and so it is worth using this device in prototypes.

Overall the significantly lower power consumption and smaller size (due to not needing peripheral components) of the Peregrine Semiconductor phase shifter makes it a better choice than the Analog Devices vector modulator. The loss in performance is not significant compared to the gains in other areas. This will be the device for which a circuit is designed and manufactured.

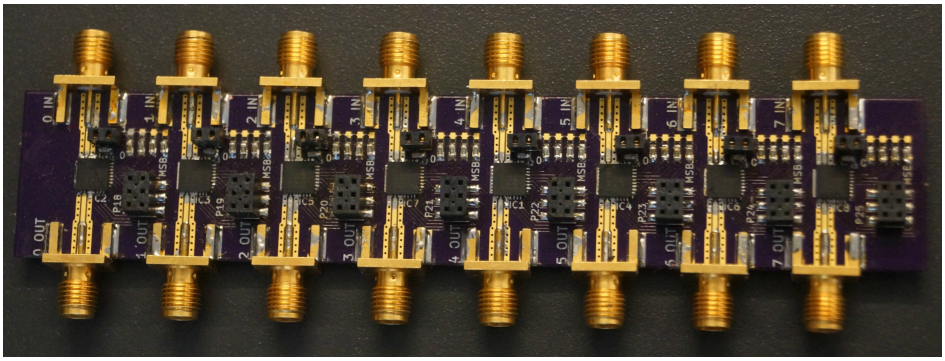


Figure 4.3: Photograph of a fully populated phase shifter board.

4.3.3 PCB design

All of the PCBs for these beamformer building blocks will be built using the same technology. First, a substrate must be chosen. The standard substrate is known as FR4. FR4 is a lossy material, and it becomes particularly problematic when the signals are high frequency and high power. In this system the signals are at 1.5 GHz. This is a moderately high frequency in terms of what FR4 can handle, but the very low power of the signals in this particular system mean that FR4 is a suitable material. Alternative materials such as Rogers substrates are available but the increased cost is not justified when FR4 is able to meet the performance requirements.

The design uses 0.8 mm FR4. This is thinner than the standard 1.6 mm substrate which will decrease losses by increasing the coupling between the signal and ground traces. It also means the signal traces can be thinner for a given impedance. All PCB designs in this Chapter were created using KiCAD EDA. Fig. 4.3 shows the completed and populated PCB.

The PCB provides eight identical channels. The signal is fed through end-launch SMA connectors, which cause fewer losses than vertical connectors. The traces between the SMA connectors and the IC are designed to be 50 Ω, although the short length of the line means losses are minimised even with a mismatched line. When moving between different width lines, neck-downs (as opposed to steps in trace width) are used to minimise reflections.

Each phase shifter is individually addressable. The address is set manually using address pins, which are pulled high or low by connecting them to power or ground respectively. Switches were created using 0 Ω resistors. There is a choice of positions in which each resistor can be soldered, one giving a logic high and the other logic low. In Fig. 4.3 all the resistors are soldered as logic low, meaning all the shifters have the same address. This was for testing purposes and resistors were later moved so that each channel had a unique address.

Each channel also has two off-board connectors - a two pin and a six pin header socket. A mezzanine board was also designed. It fitted on top of the shifter board and supplied power (through the two pin connector) and data signals to control the phase shift through the six-pin connector. The PE44280 has the ability to repeat digital signals and this was used to regenerate the data signals, which reduced the drive requirements for the microcontroller by making it only have to drive one device instead of eight. This could potentially allow for faster communications as the pulses would be better shaped if the capacitance was lower. Hence each channel has six data channels: three inputs and three outputs. The mezzanine

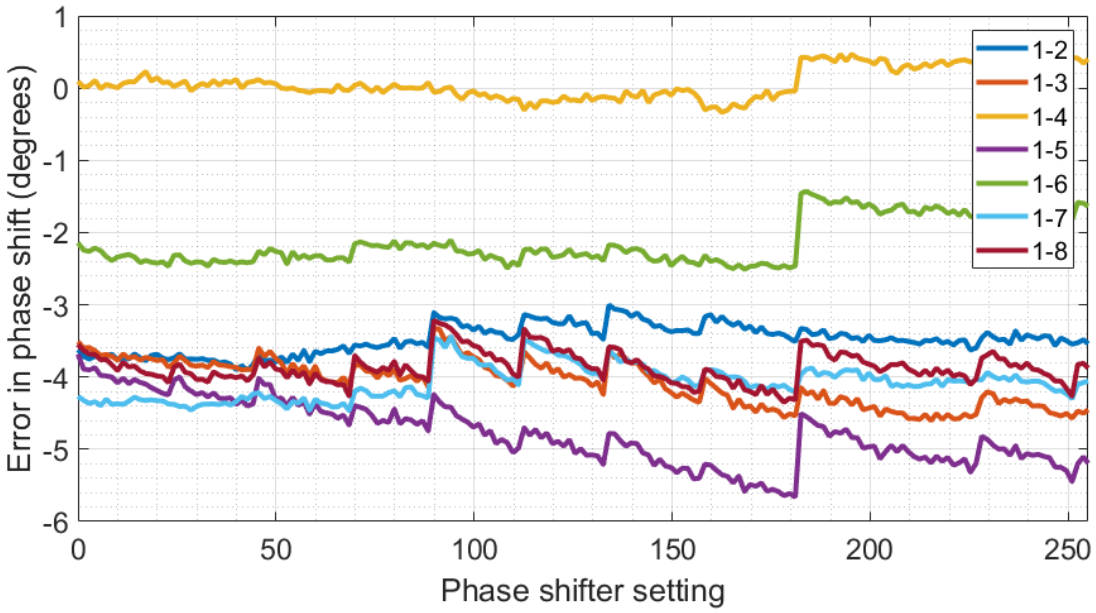


Figure 4.4: Difference in phase shift between channels, relative to channel one.

board also has a MCP1755 linear regulator for each channel, which supplied the required 3.3 V.

The decision to use a mezzanine board was taken to increase the physical distance between the RF signals and the potentially noisy digital signals. While the control signals are relatively low frequency (the maximum rate for the IC is 10 MHz), the harmonics may still be significant. The other option would be to use a multilayer PCB with ground planes to separate analogue and digital traces, but this was method was chosen for its flexibility, as well as simplicity and lower cost - four layer boards can be more complex to design and more expensive to manufacture.

The completed PCB was then tested to evaluate the performance of the device. Testing was carried out using a Copper Mountain 304/1 Vector Network Analyser.

The isolation between channels was found to be lower than the dynamic range of the VNA and so is not shown. However, the fact it was lower than what could be measured implies that it is low enough to be disregarded.

Fig. 4.4 shows the difference in phase shift of two channels that are set to the same phase shift. The difference is quoted as between channels 1 and 2, and channels 1 and 3. The error is always less than 5° and typically within a 2° degree range. As the change in beam angle is relative between channels, this error is fairly small. It could be reduced further through the use of lookup tables for each device, if they were characterised in advance. One lookup table would be relatively small as there are only 256 possible phase values.

Figs. 4.5 and 4.6 shows the result of stepping through every possible phase shift value. Both graphs have three lines plotted: 'At L1' shows the error at 1.575 GHz, the frequency of interest. 'Lowest' shows the lowest error achieved while 'Highest' shows the highest error. Fig. 4.5 gives the percentage error in phase value of one channel of the phase shifter. The percentage error is used as at large phase shifts the error could appear to be large but proportionally was still small. This does lead to a misleading impression at low values where a small error is a very large percentage. Fig. 4.6 shows that at no point was the error large.

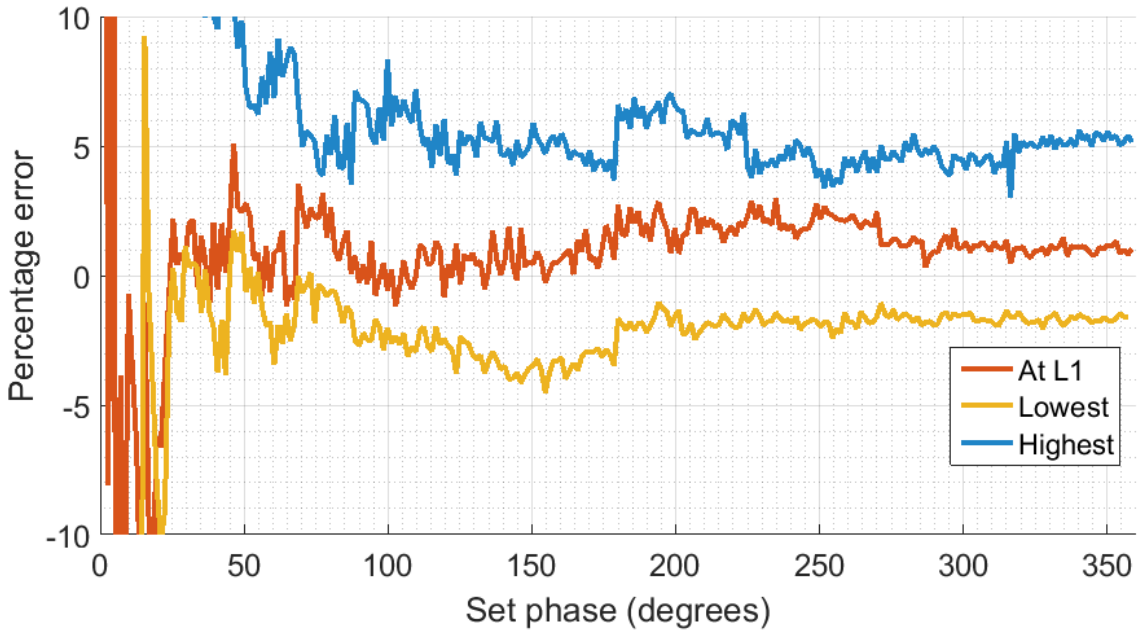


Figure 4.5: Percentage error between set position and actual position with set phase.

Hence the axes of Fig. 4.5 are restricted to show greater detail at higher phase shifts. The percentage error at the frequency of interest (labelled as L1) is low - typically less than 3% error. The other lines are the highest and lowest percentage error seen over any frequency, with the test being run between 1.5 and 1.6 GHz.

It is noticeable, particularly in the high and low traces, that there is a step around 180°. This may be to do with the optimisation of the look-up table used, which was not for the 1.5-1.6 GHz range. At 180° there will be a change in all of the bits of the phase word (transitioning from 0b01111111 to 0b10000000) and so any small systematic errors that have built up will suddenly become apparent. Modification of the lookup table for the correct frequency would significantly reduce the error.

It was found that the phase was repeatable; that is, if the phase was set to a value, set to a different value, then changed back to the original value the difference would be almost zero. The stability was also good, with no systematic drift if the phase shifter was left on the same setting for a long period of time.

The insertion loss of the phase shifter was also measured, and is plotted in Fig. 4.7. At L1 the loss was consistently within 6 dB $\pm$ 0.25 dB. The variation was periodical, implying that it is to do with the phase setting. There is no way that this can be removed with a lookup table using just the phase shifters, and so additional components (*i.e.* attenuators) would be required.

4.4 Attenuators

The decision was made to use a phase shifter instead of a vector modulator for phase shifting purposes. This decision means that if attenuation is required, this will be performed by a separate block in the beamformer. While an attenuator is not always required (particularly as the array is so small, aperture shaping is not possible) it may be needed. There are a number of devices from different manufacturers that can be used for this purpose. Ideally an

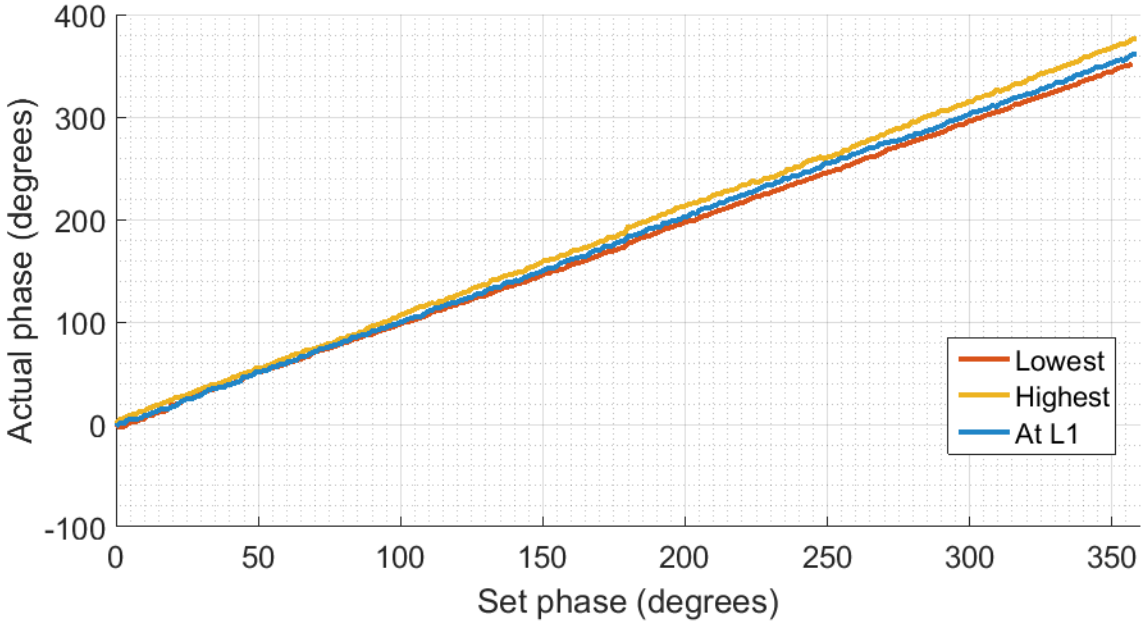


Figure 4.6: Actual phase shift value with changing set phase value.

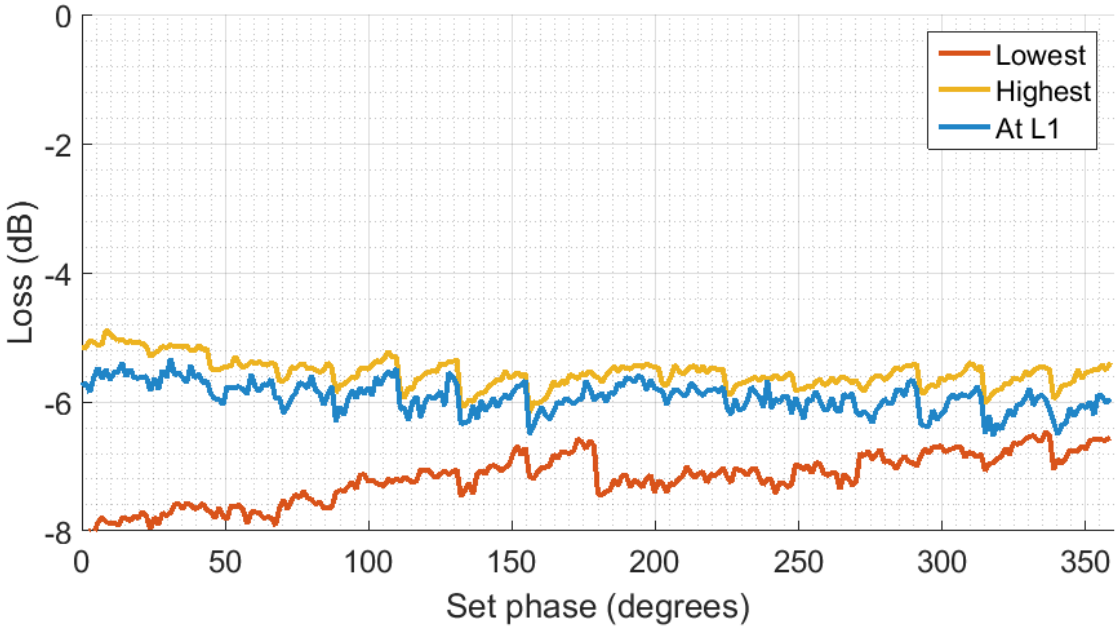


Figure 4.7: Insertion loss of one phase shifter channel as a function of changing set phase.

attenuator for this system would be low cost and have low power consumption, with small attenuation steps so that even small differences in insertion loss from other devices can be balanced out.

4.4.1 Skyworks SKY12329-362LF

This component is a GaAs (Gallium Arsenide) attenuator [93]. Internally, it has five attenuators that can be activated or bypassed depending on the digital input. It has an attenuation range between 0.5 and 15.5 dB, with an insertion loss of approximately 1.3 dB at 1.5 GHz. The device is controlled via a serial interface so that the attenuation is set digitally, with a settling time of 1.2 μs. It has a power consumption of 2.5 mW and does not need any external components to control it. As with the phase shifters the power would be supplied by a linear regulator IC, with a separate power supply for each attenuator.

4.4.2 Peregrine Semiconductor PE43712

An alternative component is the Peregrine Semiconductor PE43712 [94]. This device uses the same silicon-on-sapphire technology as the phase shifter discussed in Section 4.3.2, meaning it has a similarly low power consumption of 495 μW. This is an order of magnitude lower than the Skyworks attenuator. The PE43712 is controlled digitally, but has seven bits (compared to the five bits of the Skyworks device). The minimum step size is 0.25 dB and it has an attenuation range of 0.25-31.25 dB, which is both a wider range and in smaller steps than the Skyworks device due to the higher number of control bits. The settling time is 1.6 μs, which is slightly slower than for the Skyworks device but of the same order of magnitude. The insertion loss for the PE43712 is approximately 1.3 dB at 1.5 GHz, making it the same as the Skyworks device. In terms of cost, the PE43712 is marginally more expensive than the SKY12329-362LF, but only by a few percent (exact figures cannot be given as prices are subject to fluctuation).

4.4.3 Fixed attenuators

The final option could be to use fixed attenuators instead of digitally controlled variable attenuators [95]. These are smaller than variable attenuators and are passive devices, meaning they need no power supply and no control signals. If they were to be used, the attenuation down different paths from the antenna would be characterised and the correct value attenuator inserted. However, these devices would not be able to balance variation in attenuation caused by changing the phase, and so would have limited use. For this reason they will not be considered further.

Overall both active devices have similar performance, cost, and power consumption. More importantly, combining one of these devices with the PE44820 digital phase shifter still results in a solution with significantly lower power consumption than the AD8341 vector modulator, proving that it is not a suitable solution for this application. However, there was no requirement for these devices in the first system prototype so no PCB has been designed. If it were needed, with care the layout for a PE43712 board could be designed so that the base

board was compatible with the mezzanine board for the PE44820 phase shifters, reducing costs and manufacturing time.

4.5 0° Splitter/combiners

Another key building block of beamformers for array antennas is one that gives the ability to combine signals. Alternatively it might be necessary to split a signal so that it can be fed down two separate paths. Typically both of these tasks can be carried out by one component.

The key requirements for a splitter/combiner are for there to be a consistent phase shift and attenuation down the two paths, and for the isolation between the two paths to be good.

4.5.1 Microstrip power dividers

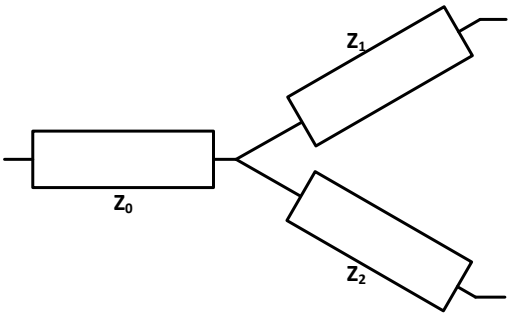
There are three main ways of creating a microstrip power divider. They are illustrated in Fig. 4.8. A T-junction power divider is the most basic power divider that can be made, whether it is from waveguide or microstrip. While it can be made lossless, it has a number of disadvantages, including not being matched at all ports and also the output ports are not isolated from each other. A resistive divider inserts lumped element resistors to match the outputs, but the ports are still not isolated from each other. A Wilkinson power divider is not lossless, but it is isolated. A resistive divider places the resistors in series with each port, while a Wilkinson power divider places a resistor between the two output ports. The device is therefore not lossless but the losses are dissipated through the resistor, meaning there are no reflections and the output ports are isolated [96]. In addition, to prevent reflections at the point where the outputs join the input, there is a $\lambda/4$ portion of microstrip with an impedance of $\sqrt{2}Z_0$.

The main disadvantage of all of these designs is the fact they are made from microstrip and can therefore be quite large. This is more of a problem at lower frequencies (like GPS L1) as the devices scale with the wavelength, and so a large wavelength can mean a large feature on a circuit. For a Wilkinson divider at GPS L1 (approximately 1.575 GHz) the $\lambda/4$ section would be roughly 45 mm long. Monolithic devices can be used to minimise the size of the circuit.

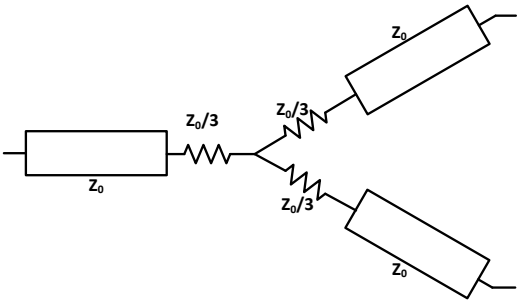
4.5.2 Mini-circuits SCN-2-19+

The Mini-circuits SCN-2-19+ is an ultra-small ceramic power splitter [97]. It acts as the power splitter section of the Wilkinson power divider, removing the need for a long, lower impedance section. It still requires the external isolating resistor but the device is significantly smaller than the footprint required for a fully-microstrip power divider.

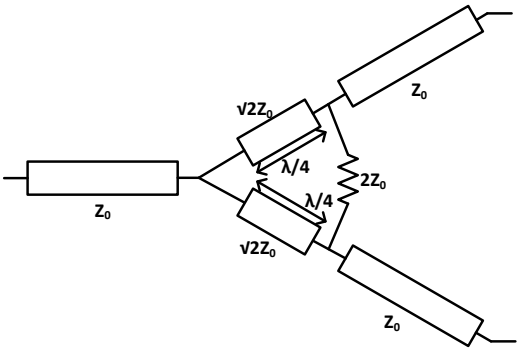
The SCN-2-19+ requires no power supply (it is a passive device) and the only external component is the isolating resistor. The device is low cost, so the increased cost of components (compared to microstrip) is mostly offset by the reduction in cost from having a smaller circuit board. It is also wider band (typically 250 MHz) as there are fewer paths with lengths or widths dictated by the frequency, making this choice more suitable if the design is to be adapted to dual band detection (for instance, detecting jamming on L2 at 1.227 GHz, or detecting other constellations).



(a) T junction power splitter is matched at all ports but is not lossless and the ports are not isolated. $1/Z_1 + 1/Z_2 = 1/Z_0$.



(b) Resistive power divider. It is lossy and the ports are not isolated, but it can be matched.



(c) Wilkinson coupler. It has matched ports and they are isolated; however, it is not completely lossless.

Figure 4.8: Different types of microstrip power couplers.

4.5.3 PCB design

Fig. 4.9 shows the completed PCB for this device. This PCB does not have an associated mezzanine board, unlike the phase shifter board, as no power supply or digital control is required. The channels were arranged to minimise the space required, but also so that they could easily be cascaded. By connecting together multiple splitter/combiner circuits in series, it would be possible to combine four or more signals.

The connections for each channel were labelled on the board. The two input ports are labelled x_1 and x_2, and the output is port 3, in the discussion below. All four channels were measured and found to be similar so the results are only given for one channel. Measurements are taken between the two connectors, meaning the stripline between the connectors and the splitter devices is also included in the measurement (this also applies for all other measurements of PCBs in this Chapter).

Fig. 4.10 shows that the phase shift is almost the same for S_{13} and S_{23}, and does not deviate with frequency. This is in accordance with the datasheet, which predicts a phase imbalance at room temperature and at GPS L1 of less than 0.5°. This means that the phase shift through either path is not significantly different and will not need to be corrected for by the phase shifters, simplifying the control of the system.

Fig. 4.11 shows the loss for the two outputs. A 3 dB loss is expected as 3 dB equates to a loss of half of the power. The output power on one port is half of the input because the input signal has been split in half to be sent to the two outputs. The graph shows that at 1.5-1.6 GHz the loss is around 3.5 dB, meaning 0.5 dB has been lost in addition to the expected

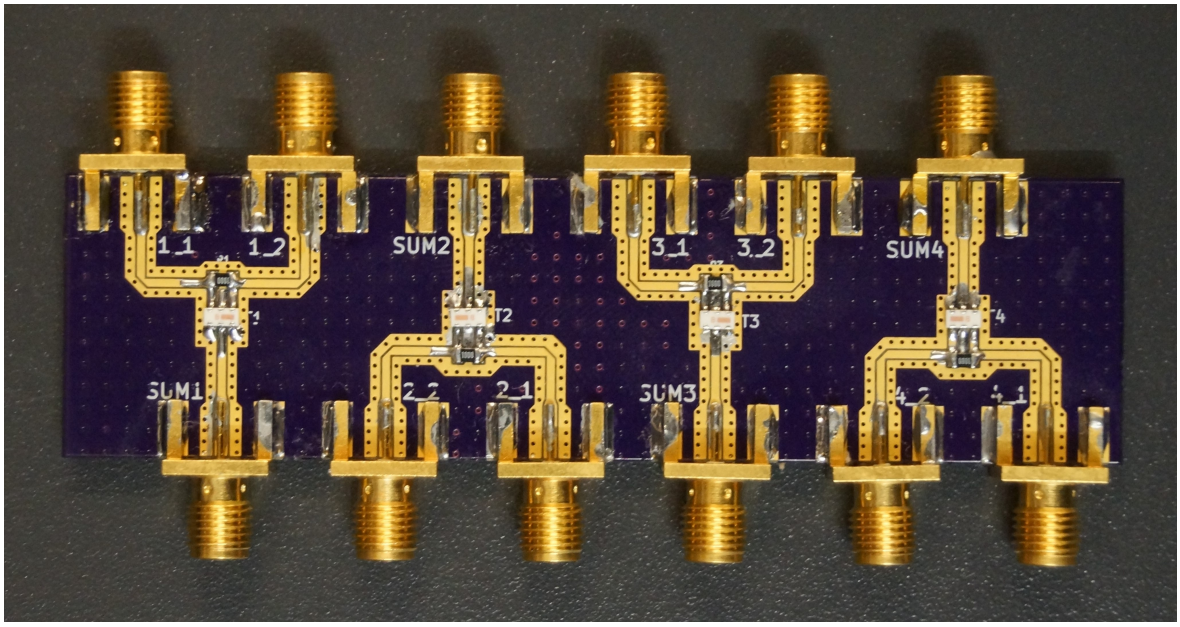


Figure 4.9: Final PCB design for the 0° splitter-combiners, MiniCircuits SCN-2-19+.

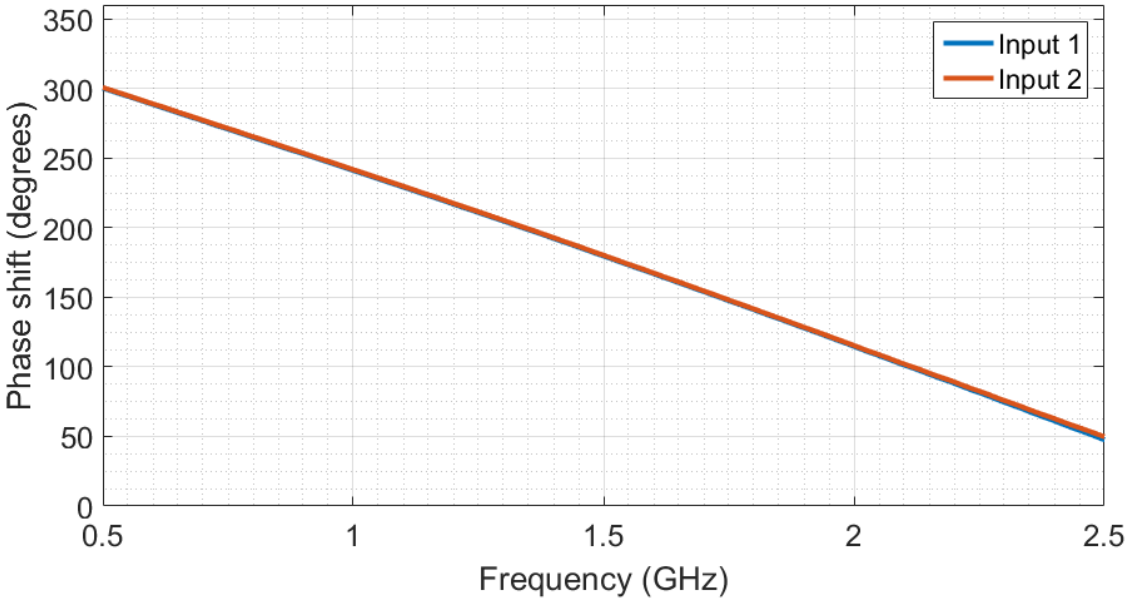


Figure 4.10: Phase shift of S_{31} and S_{32} for one channel of the splitter PCB. The input ports are 1 and 2, and the output is port 3.

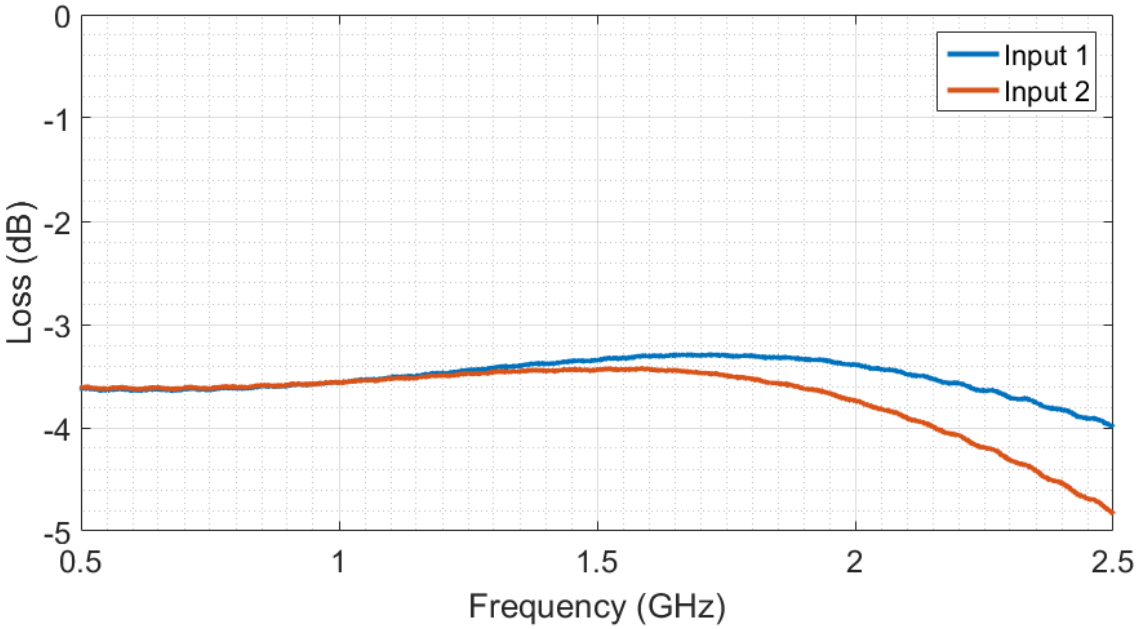


Figure 4.11: Loss in S_{31} and S_{32} for one channel of the splitter PCB. The input ports are 1 and 2, and the output is port 3.

loss from splitting the signal. The loss is also slightly imbalanced, with one output having a slightly higher loss than the other. This is not a fault of the PCB design; inspection of the datasheet reveals that the loss for the two outputs varies with frequency and at 1.5-1.6 GHz there is a discrepancy of slightly more than 0.1 dB, just as is seen with this PCB. Therefore the circuit design and manufacture has not impaired the operation of the device, and all parameters are within acceptable values in that they will allow the whole system to function.

Fig. 4.12 shows the isolation between outputs for all four channels. The isolation is at a maximum at GPS L1. This will be due to the traces having their impedance matched at 1.575 GHz; at other frequencies the radiation from the traces will be higher. The isolation remains low (below -30 dB) over a 25 MHz band, and below -25 dB over a 50 MHz band.

4.6 Digital switches

In addition to splitters and combiners, switches are often needed to create beamformers. As with previous building blocks in this Chapter, the focus will be on monolithic ICs that can be mounted on a PCB as it has been demonstrated that they provide the best solution in terms of cost and size.

It is not known how many inputs will be required for each switch, so for flexibility a four-way switch will be designed. The price difference between a two-way and a four-way switch is normally small, so this decision adds a lot of flexibility with little additional cost.

In this Section, 'input' will refer to a signal that can be selected, while 'output' refers to the common connection. These devices are actually reversible so the signal can flow in either direction, but this convention will be used for simplicity.

A good switch is one that can be changed rapidly. It should have a low insertion loss when in the on state, but when an input is not selected it should be correctly terminated internally so that reflections are minimised.

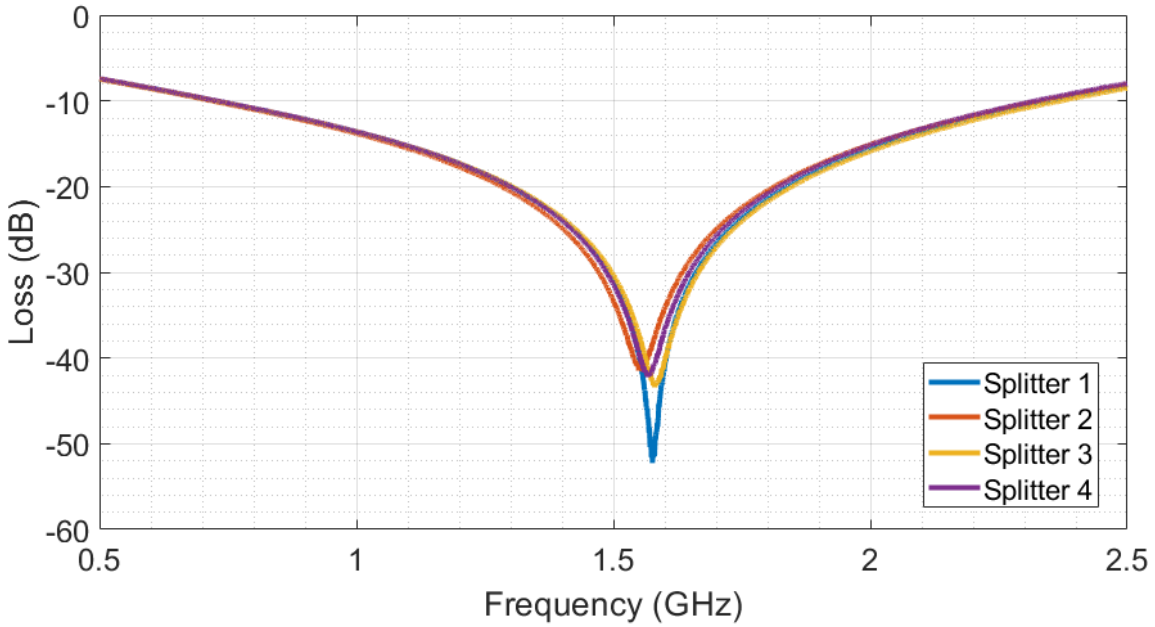


Figure 4.12: Isolation between output ports for all four channels of the splitter/combiner board.

4.6.1 Peregrine Semiconductor PE42442

Once again, Peregrine Semiconductor provide a component which shows good promise. The PE42442 is a four-way, digitally controlled RF switch that operates between 30 MHz and 6 GHz [98]. Control is via a three-pin parallel connection, although it can be modified to be two pin if the 'all off' state is not required. This control scheme requires more wires (and therefore more pins from a controller) than the addressable serial connection used by the PE44820 serial communications scheme.

It has the low power consumption which is a characteristic of Silicon-on-Insulator devices, using just 0.36 mW. It also has excellent isolation, with the coupling between channels typically being less than -54 dB. The insertion loss is low (0.9-1.1 dB). The return loss varies depending on whether the input is selected or not: with the input selected the return loss is -22 dB, but if it is not selected (and instead terminated inside the device) the return loss is only -17 dB. The switching time is relatively slow, at 255 ns, and the maximum switching frequency is 25 kHz. Both the switching frequency and power consumption can be improved if an additional (negative) power supply is used. However, this would use more space, and have a higher bill of materials cost.

4.6.2 Analog Devices HMC345

The HMC345 is a GaAs four way switch made by Analog Devices (formerly Hittite) [99]. Like the PE42442, it is also controlled via a parallel two wire interface (there is no 'all off' setting, which reduces the flexibility of the device slightly but is not actually required in this work). It operates between DC (0 Hz) and 8 GHz, which is a wider range than the PE42442. However, both devices have a range of operation far wider than is necessary, so can be considered equal in that regard.

In terms of RF performance, this device is slightly worse than the PE42442. The insertion loss is 1.7 dB, the isolation between inputs is 42 dB and the return loss is 14 dB for deselected

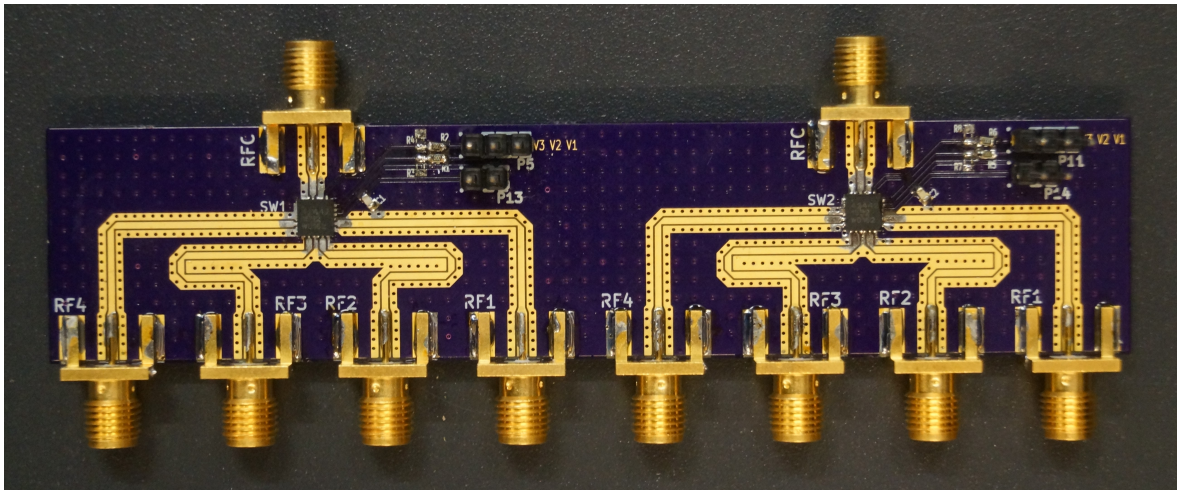


Figure 4.13: Final, populated PCB for the PE42442 digital switches.

inputs, and 16 dB for the selected input. The power consumption is also higher, at 12.5 mW. On the other hand, the switching speed is faster with a settling time of just 40 ns. The main disadvantage of this device is its cost, which is almost an order of magnitude higher than the Peregrine Semiconductor device.

Another disadvantage of this device, certainly for this application, is the pin layout. The device has the control signals in amongst the RF inputs. The board will be designed so that inputs are on one side of a narrow PCB and outputs are on the other side. This layout means the digital control signals are forced to be close to the RF traces, risking noise coupling from the digital signals to the RF inputs.

Overall the PE42442 was chosen. Its slower switching speed should not be a problem as the settling time for the phase shifters is comparable and so the switches will not cause a bottleneck. This slight disadvantage (that may not cause any problems) is offset by the lower power consumption and cost, and better layout.

(Note: Analog Devices have since released their own Silicon-on-Insulator technology, and the HMC7992 uses this process. The specifications, particularly power consumption, are significantly better than the HMC345 and comparable to the PE42442. However, at the time this work was being carried out this device was not available.)

4.6.3 PCB design

Fig. 4.13 shows a populated PCB that carries two switches. The four inputs are on one side of the board with the common output opposite. Control signals are kept away from the RF signals as much as is feasible. Power and control signals are fed to the IC via a mezzanine board, which connects through a two pin and a three pin connector to each IC. The path length of each trace is matched so that no phase shift compensation is required. The mezzanine board carries one power supply IC for each switch channel, as well as connectors for the switches.

Fig. 4.14 shows the insertion loss of the device. It was measured by connecting the VNA to one input and the common output, and selecting the input that is connected. Hence the loss associated with a signal passing through the device (theoretically unimpeded) is measured. The insertion loss was almost identical for all four input channels, at around

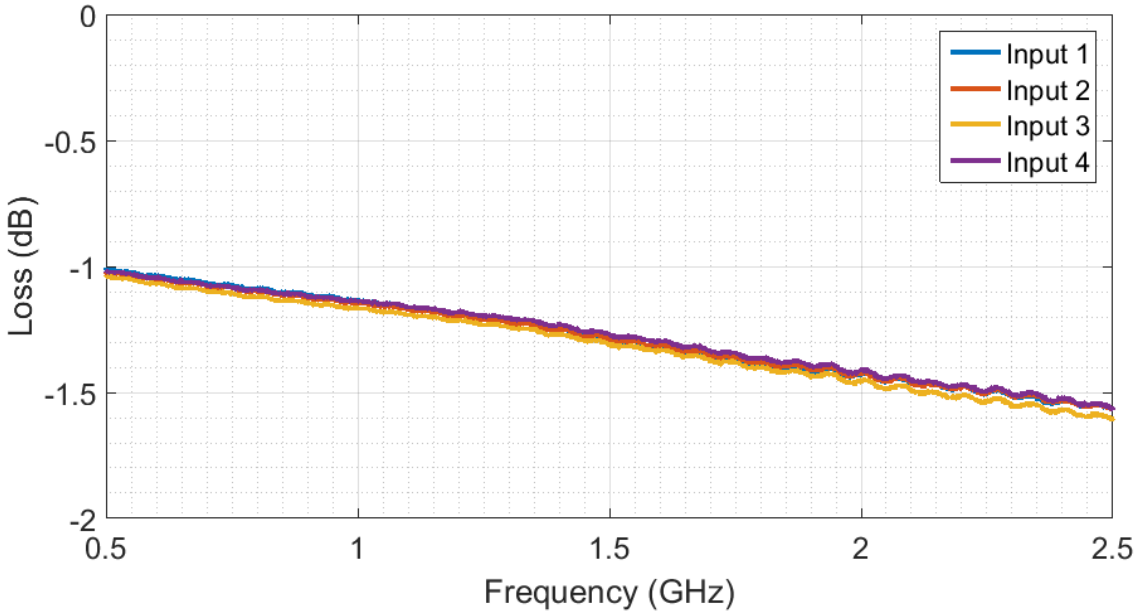


Figure 4.14: The loss through the switches when in the on state.

1.3 dB at 1.5 GHz. This is slightly worse than the value given by the datasheet; the difference is most likely due to losses in discontinuities such as the connector meeting the board, the trace changing width near the IC, and the IC meeting with the trace.

Fig. 4.15 shows the isolation of one channel of the switch. The measurements were taken as follows: port one of the VNA was connected to the fourth input of the switch, and port two to the common output. The three other inputs were then selected in turn. Thus if the isolation between two input channels was poor, signal from the VNA on input four will couple into the input selected by the switch and appear at the output. The isolation was always lower than -48 dB for all inputs. However, when input three is selected the isolation is significantly higher than for the other two inputs. According to the datasheet the isolation between inputs is between 54 and 61 dB at 1.5 GHz, which is what is seen with two of the traces. When the value is worse for the third trace, it is likely due to some factor outside of the IC. Inspecting the PCB design as shown in Fig. 4.13 shows that ports three and four are adjacent on the device and the tracks run parallel to each other for some distance. Thus it can be concluded that there is some coupling between traces on the board. As the design shows mirror symmetry it can be assumed that inputs 1 and 2 will also have some coupling. However, the amount is low enough that the board does not need to be redesigned immediately.

Fig. 4.16 shows the phase shift through the board. The lines are given as the phase difference relative to input one; the absolute phase difference between the input and output is unimportant and only the relative difference matters. The graph shows that between symmetrical channels (one and four, and two and three) the difference in phase is very small. However, the difference is larger between the 'outside' channels (one and four) and the 'inner' channels (two and three), at around 5-6°. Considering Fig. 4.13, the mismatch can be explained. The board was designed so all paths were all equal length. Due to geometrical constraints, the inner channels had more corners than the outside channels. An assumption was made on calculating the phase shift around corners, based on information from Microwaves

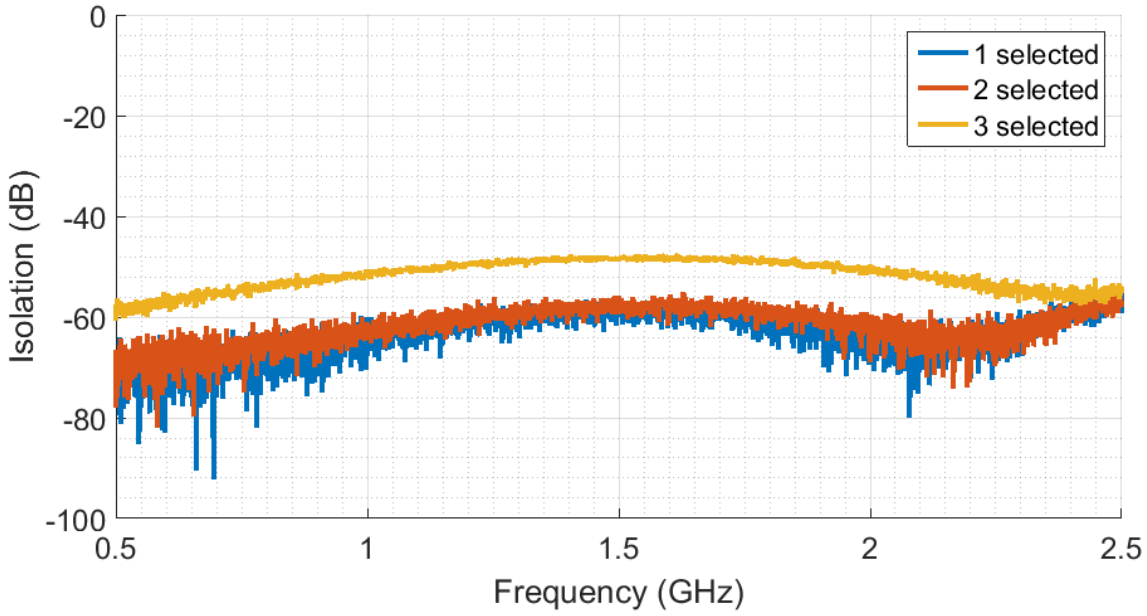


Figure 4.15: Comparison of isolation through different channels of one PE42442 digital switch. The VNA was connected to input 4 and the common port. The other three inputs were then selected in turn.

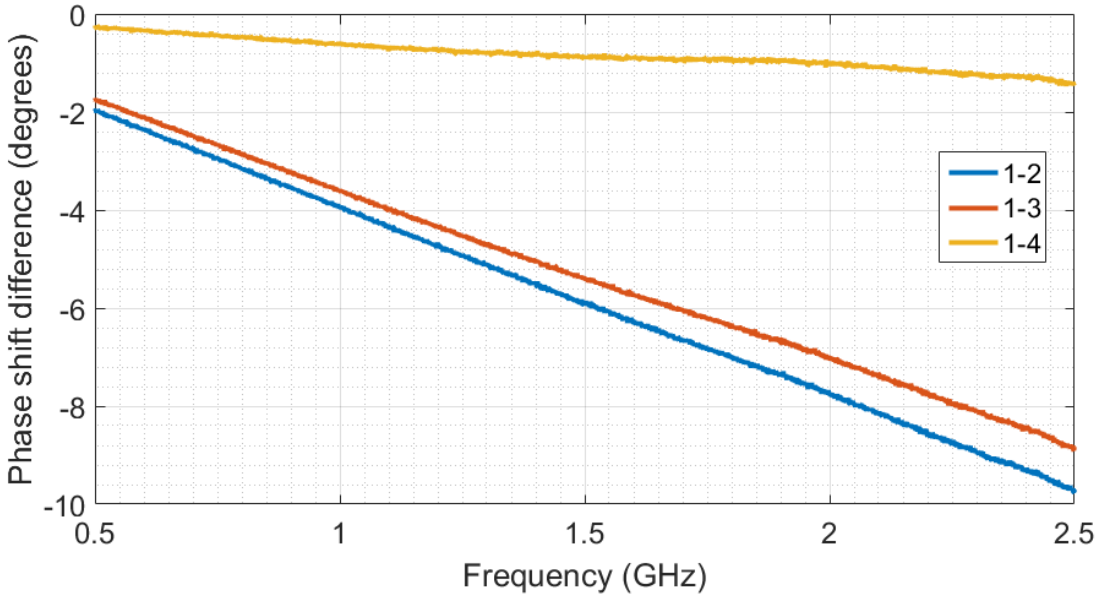


Figure 4.16: The phase shift through the switches when in the on state. The values are relative, given using channel one as a reference. The phase shift through channels one and four is the same, and two and three are approximately the same, although the two pairs are different from one another.

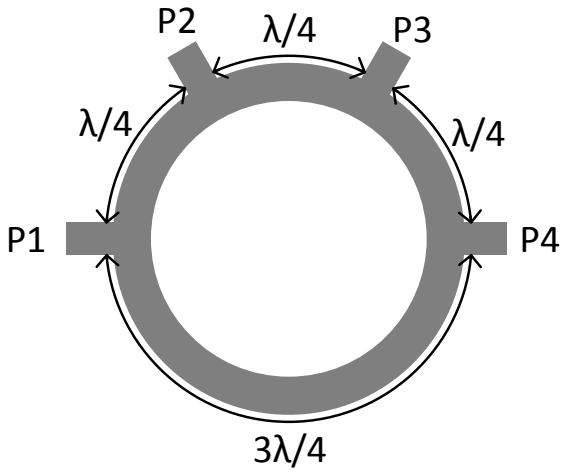


Figure 4.17: Diagram of a rat race hybrid coupler.

101 [100], which states that around a mitred bend the effective length is equal to half the width of the trace. This assumption appears to not be quite correct, or there are extra effects at play, as the traces with more corners have a longer delay than those with fewer corners. This problem can be corrected using the phase shifters if necessary. Future iterations will include more detailed simulations now that it is known that assumptions cannot be relied upon.

4.7 Hybrid coupler

The 180° hybrid coupler is an important part of this design. It was established in Chapter 2 that the best method of using a small, low cost array to determine the angle of arrival of a signal is to use the null produced by the difference of two beams. It was also discussed how using the null alone could be ambiguous. Hence the signal produced by summing two pairs of elements of the array is also used. To speed up the process it is useful to be able to produce both sum and difference beams at the same time; otherwise the two patterns would have to be produced by using the phase shifters. This would not only slow the process, but it would cause noise and inaccuracy in the signal due to the time difference between the two readings. Hence some method of combining the signals so that both the sum, Σ, and difference, Δ, beams can be produced at the same time.

4.7.1 Rat race

The traditional way of creating both signals is to use a microstrip feature known as a rat race. An example is shown in Fig. 4.17. The inputs from the array are fed into ports P1 and P3. Both inputs have the same path length of $\lambda/4$ between their input and P2. This output therefore produces a signal where the two outputs are in phase. This is Σ. In contrast, between P1 and P4 the path length is $3\lambda/4$, while the path between P3 and P4 is just $\lambda/4$. Hence there is a path length difference of $\lambda/2$, which equates to a phase difference of 180°. This phase shift is exactly what is required to add the two pairs of antennas in antiphase to create the Δ beam.

The main disadvantage of this design is its size. The path lengths have to have the

specified path lengths, which are dictated by the wavelength. The overall circumference is 1.5λ, which at GPS L1 is almost 30 cm (the wavelength is approximately 19.1 cm). Hence the diameter of the whole feature will be 9.1 cm across, which is not good for this system. Another, more minor, disadvantage is the points at which the ports connect to the rat race. The connections are typically T-junction power dividers, which will cause reflections and be inefficient.

4.7.2 QCN-19+

In Section 4.5, it was discussed how a microstrip power splitter can be replaced with a much smaller device. The SCN-2-19+ discussed in that Section has a 0° phase shift between the input and the two outputs. This could be used to improve the connections to a rat race. However, the same company, Mini-circuits, also make a quadrature splitter, which is far more useful in this situation. This quadrature splitter, part number QCN-19+ [101], has one input and two outputs (although as with the SCN-2-19+ device, the device is reciprocal and can be used either way round). The signal input to port one is split in half. Half of the power is fed to port two with no phase shift. The other half is fed to port three with a phase shift of 90° relative to port two. This device does not require an external isolating resistor.

By connecting together four QCN-19+ devices, it is possible to create a rat race that can be as small as required. As long as the path lengths between devices are matched, there are no constraints on the physical lengths of traces. This is the method that will be used in this work; however, details will remain sparse as this is the Intellectual Property of Dr. Robert Watson.

4.8 Making a whole system

At this point an array, an antenna and now a beamformer have been designed. This is sufficient to create a complete system for real world tests.

4.8.1 Antenna

As the antenna design discussed in Chapter 3 did not result in a working design, henceforth the original design made using FR4 and pictured in Chapter 2 was used. The array was also simplified, so only two elements were used, as shown in Fig. 4.18. Individual antennas were fabricated on squares of FR4 with each side being 9.5 cm. The antenna was printed on one side of the board, and the other side had a continuous ground plane covering the whole area of the board. This allowed the antenna array to be redesigned if necessary. It would be easy to rearrange the antennas in the field as two tiles next to each other would give the correct inter-element spacing.

4.8.2 Steering the beam

As the array design had been simplified, the beamformer could also be made simpler. Fig. 4.19 shows the new, simplified beamformer. It does not use any splitters or switches.

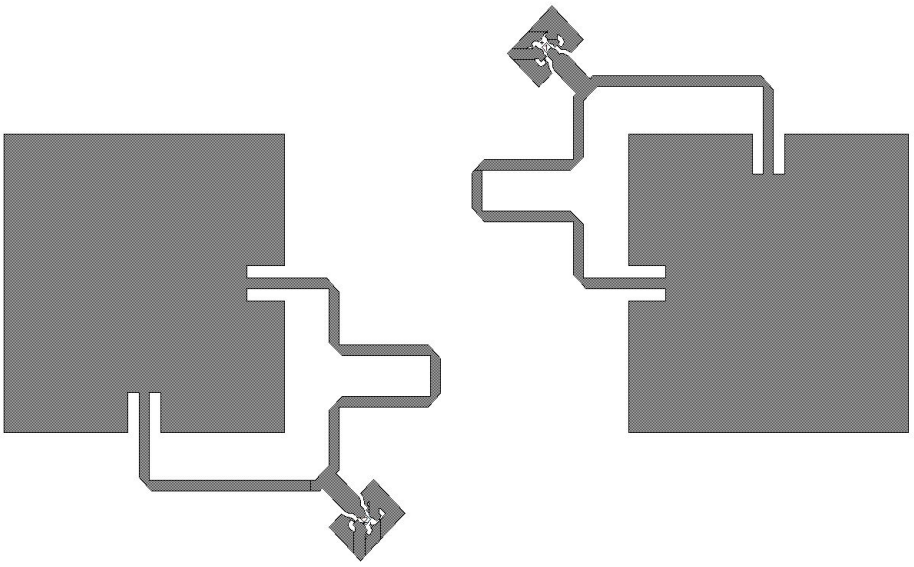


Figure 4.18: Two element antenna array used for preliminary tests.

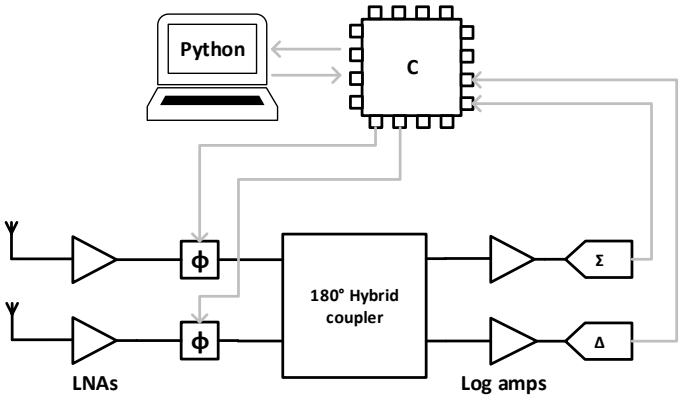


Figure 4.19: System diagram for the setup for exhaustive sweeps.

Between the antennas and the phase shifters are a series of Low Noise Amplifiers (LNAs). The overall gain was approximately 32 dB, in two stages. The devices used are MGA 62563 LNAs [102]. These have a high power consumption of approximately 80 mA per device and so are not particularly suited for a UAV-based system. However, their benefit is a very high dynamic range, meaning that they are not saturated by strong signals and would still be able to receive true GPS signals in the presence of jamming. This would be advantageous to a system designed for mitigation. For the final design the exact LNA used may change but for this prototype, the high power consumption devices will not affect operation.

Fig. 4.19 makes reference to 'log amps'. These are logarithmic amplifiers. The devices used are LT5534 RF power detectors [103]. They produce a DC voltage that is proportional to the power contained in the RF signal that is their input. Hence they convert the (still carrier frequency) Σ and Δ signals down to a DC signal that can be read by an Analogue-to-Digital Converter (ADC).

The system is controlled by a laptop running Python. Python is slow to run (as it is an interpreted language, meaning it is compiled at run-time) but it has simple-to-use libraries for important functions like USB serial communications. For these reasons it was chosen for this prototyping stage of the work.

The laptop is connected via a USB serial connection to a microcontroller, which in this case was an NXP LPC1549 on an mbed development board, part number OM13056 [104]. While it would have been possible to use a USB or ethernet switch to produce SPI and parallel signals to control the beamformer, this method would be slow. Modern microcontrollers have many built-in peripherals such as SPI modules, making them simple to use. Another common peripheral on microcontrollers is an ADC module. This has the benefit of reducing the number of components required. A microcontroller with an ADC is able to record the values of Σ and Δ and report them back to the PC with no additional components required. In this case the LPC1549 device used has built in 12-bit, 2 Msps (Megasamples per second) ADCs and these were used for recording the Σ and Δ signals.

The Python program functions as follows. At the start of the program's operation, the phase shift to one element is fixed at 180°, which is halfway through the full span of a sweep. The phase shift of the other element is then set to zero, shifting the beam as close to endfire as possible. The values of Σ and Δ are recorded for these settings. The Python program then incrementally steps through every possible phase setting of the second phase shifter, with the first phase shifter remaining fixed at 180°. For all phase shift settings, a lookup table optimised for lower frequencies was used. For each point the powers of Σ and Δ are recorded by taking the average of 100 ADC readings. The ADC readings for both signals, as well as the phase settings, are recorded in a .csv file for processing.

The microcontroller controls the beamformer throughout this process. Commands are sent via the USB serial connection to an FTDI device, which converts them to UART signals read by the microcontroller. The commands are fixed length and contain a device type, an address, and a value. For instance the device type might be a phase shifter, the address is phase shifter one, and the value is a binary value between 0 and 255, corresponding to the desired phase shift. An additional command (with zeroes to pad the message to the correct length) is used to instruct the microcontroller to perform a read of Σ and Δ. When the

Figure 4.20: Photograph of the typical landscape found at the Sennybridge Training Area. Photograph reproduced with the permission of the Commandant at Sennybridge.

microcontroller receives an incoming message it will discard any bytes until it receives an 'H', 'W', 'A' or 'R' (sHifter, sWitch, Attenuator, or Read), meaning partial messages received during startup are discarded.

Manipulating strings to make them a fixed length is easy in Python, but more involved in C, which is what was used to program the microcontroller. As a result when the microcontroller performs a read the return message (containing the ADC readings) does not have a fixed length. Instead it is two decimal numbers between one and four digits long, separated by a space. At the end an arbitrary non-numeric character (in this case a 'B' was used) indicates the end of the message. Hence the Python program is able to determine when it has received one complete message and Σ and Δ values will not be mixed up. Both of the methods used for synchronisation of the two systems are imperfect and stalls do occur on startup, but it is an acceptable fault level for a proof of concept.

4.8.3 Testing scenario

To find out if a system can really detect a jammer, it is best to test it in a real scenario. Tests took place at the Sennybridge Training Area in the Brecon Beacons, Wales, UK. GPS jamming is illegal under the Wireless Telegraphy Act 2006 [105], but as this location is a military training site, permission can be obtained to jam. It is a relatively open area with few features such as buildings or large plants, meaning reflections are kept to a minimum. Fig. 4.20 is a photograph taken at the test site in Sennybridge.

Fig. 4.21 shows the jammer used in tests. The conditions of being allowed to jam GPS at Sennybridge require all jammers to be modified so that they only jam signals within a 20 MHz band centred on GPS L1, so while the jammer is a 'real' jammer it has been modified slightly. However, its output (shown in Fig. 4.22) is representative of the sort of jammer that would be found by law enforcement. This jammer has an output of around 300 mW, with a sawtooth frequency sweep within the allowed band. It was not designed to jam any systems other than GPS.

Figure 4.21: Jammer used in tests.

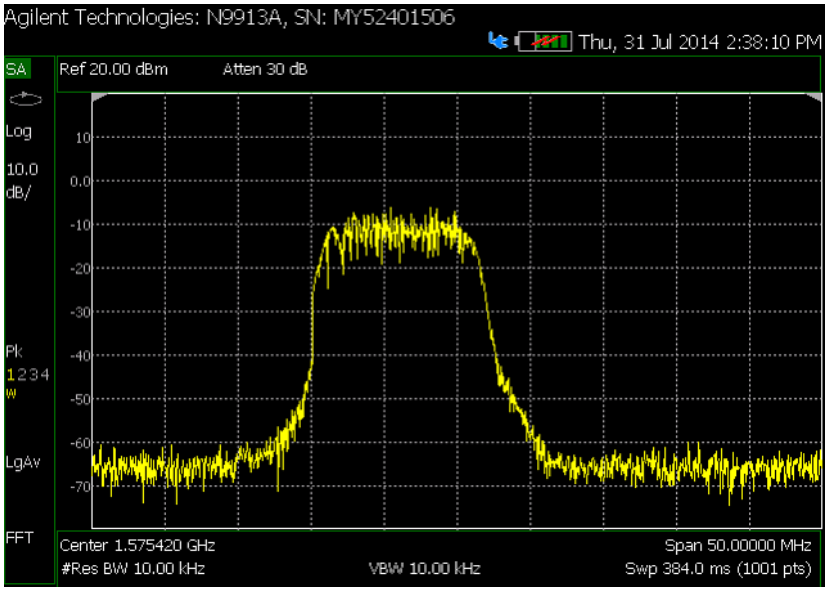


Figure 4.22: Spectrum of Jammer 2, as measured by an Agilent FieldFox spectrum analyser.

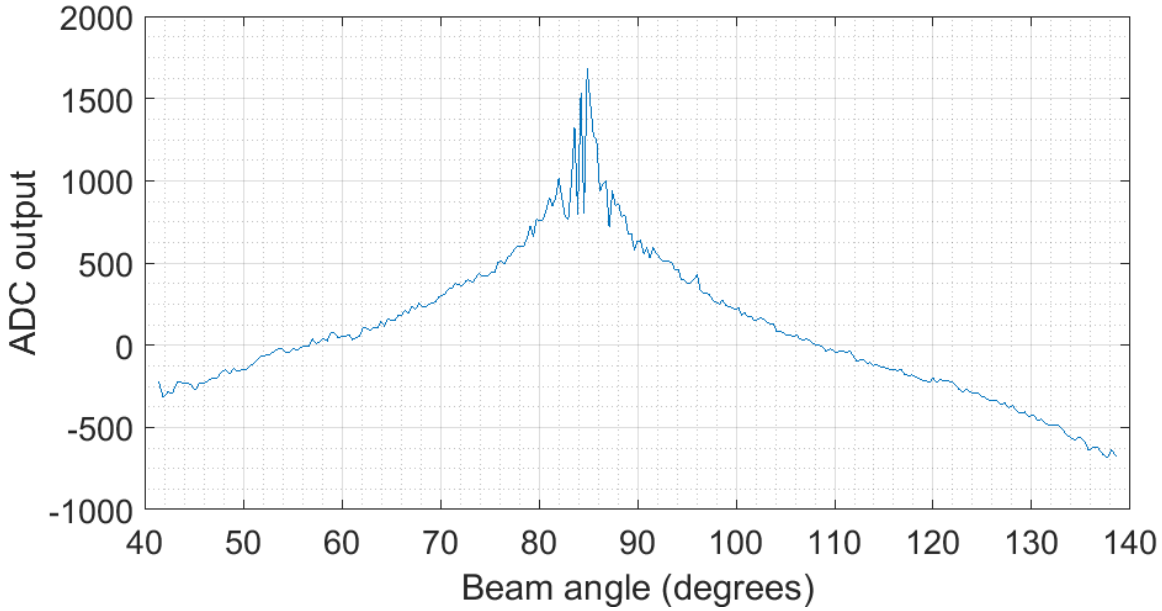


Figure 4.23: Results of holding a jammer directly in front of the array.

4.8.4 Results

The first test placed the jammer roughly in front of the array, at a distance of approximately 10 m. The jammer was held to be in the same plane as the receiver antenna. A single sweep was performed and the results saved. The results are shown in Fig. 4.23. This graph (and all subsequent ones) plot the beam angle relative to endfire against the signal strength. The signal strength is recorded in ADC counts. The ADCs used are 12-bit, meaning the output can range between 0 and 4095. This number is proportional to the voltage produced by the logarithmic amplifiers, and therefore proportional to the log of the power of the signal. However, the ADCs have not been calibrated so this value cannot be turned into a power level directly.

The values plotted in Fig. 4.23 are the result of $\Sigma - \Delta$. As expected, when the beam (in this case, both Σ and Δ beams) are pointed away from the jammer the overall signal level is low. It is negative as the value of Δ is slightly higher than Σ away from the centre of the beam pattern (see Fig. 2.10 in Chapter 2). As the beam is swept past the jammer, which is at around 85° relative to the endfire of the array in this test, the signal reaches a maximum. The null of the Δ beam is pointed directly at the jammer at the same time as the peak gain of the Σ signal. Once the beam has passed the jammer, the signal level drops away again.

This result shows that the system is working as intended. Using just two elements of an array has produced a signal with a clear peak to indicate the direction of the jammer. The next tests prove the system's ability to determine the angle of arrival when it is away from broadside.

Fig. 4.24 shows the result when the jammer was held to the left of the array. The jammer was the same as the previous test, and it was the same distance from the receiver.

Comparing Figs. 4.23 and 4.24 highlights a number of things. The jammer, and the distance between it and the receiver, were the same for both experiments. However, the peak magnitude is lower (approximately 1200, compared to 1700 when the antenna was broadside to the array). The peak is also wider, making the exact angle of the jammer harder to

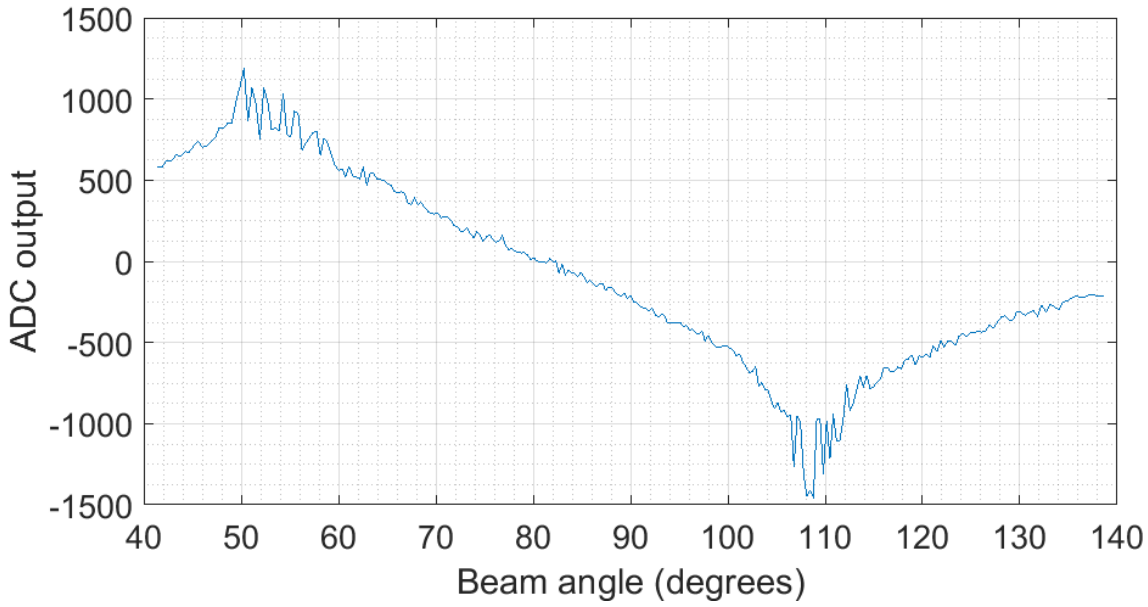


Figure 4.24: Results of a sweep when a jammer was held to one side of the array.

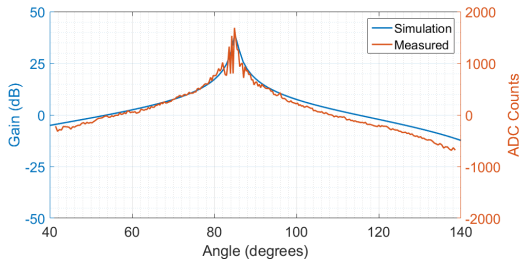


Figure 4.25: Sweep from Fig. 4.23, overlaid with a simulation of $\Sigma - \Delta$.

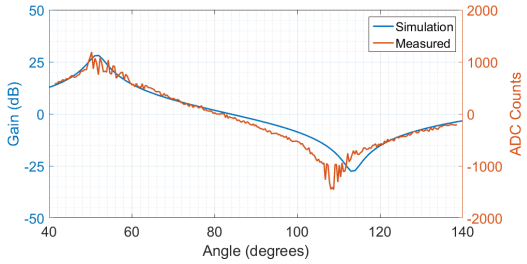


Figure 4.26: Sweep from Fig. 4.24, overlaid with a simulation of $\Sigma - \Delta$.

pinpoint. When the jammer is held far to one side, a minimum is also present. This is due to the link between the Σ and Δ beams, where Δ is simply Σ with an additional 180° phase shift applied to one element.

Figs. 4.25 and 4.26 show the two sweep results, overlaid with a simulation of the array. The array simulations were chosen by finding the phase shifter settings for the highest peak of the sweep result, then putting these into the AWR simulation. A small amount of tweaking was required to line them up exactly. This will be due to inaccuracies through the beamformer. The height was also adjusted to make the lines the same position on the y axes - this can be justified as the ADC counts axis has not been calibrated.

The overlaid graphs show that the difference observed between the two tests is exactly what would be expected. The smaller aperture size when the system is searching to the side decreases the peak of the Σ beam, as well as increasing the width of the null of the Δ beam. This decreases the precision and sensitivity of the system, as is seen in the results.

While the system was able to find the jammer in different positions, there are a number of ways it can be improved. The first problem is that the sweep is very slow. There is a communications bottleneck between the PC and the ARM microcontroller slowing down the speed at which the sweep can occur. A one dimensional sweep takes approximately 13 seconds, which is not fast enough for real time tracking. It is worth noting that in Fig. 4.26, there is a discrepancy between the simulated and actual results. As the beam steers away

from broadside the angle between the main beam and the first null will vary according to (4.1), where D is the aperture and λ is the wavelength. The discrepancy arises due to coupling between antenna elements, which changes the effective aperture of the antenna.

$$\Delta \cos(\phi) = \frac{2D}{\lambda} \tag{4.1}$$

The second problem is also related to speed. The tests show that in a realistic scenario there is one peak, and away from this the signal is low. However, this method of performing an exhaustive sweep tests many points that are nowhere near the actual area of interest. The system is inefficient in that it tests all positions, whether or not they show potential.

The final problem with this method is there is currently no way for the system to determine if the peak has been found. The only method would be to sweep through the data after it had been recorded, saving the largest value of $\Sigma - \Delta$ as the solution.

4.9 Conclusion

In this Chapter a number of important building blocks were identified as common in phased array antenna beamformers. Components were chosen to create each of these building blocks. Each component was made on a separate board so that they could be rearranged as necessary. A simplified antenna and its associated beamformer were then constructed. Python code was written to control the system via an ARM microcontroller. Tests were run using a real jammer in a realistic environment and it was found that the system could indeed be used to determine the direction of arrival of a GPS jamming signal.

Chapter 5

Algorithms for efficient searching

In which: Different types of algorithms are studied . . . Two algorithms are tested . . . An algorithm is chosen for real world tests . . . Results are presented

In this Chapter the software that performs the search was improved. It was shown in Chapter 4 that the hardware is capable of using a phased array with an exhaustive sweep to locate an emitter. However, this is very inefficient due to the fact the algorithm tests every single possible point. The shape of the beam pattern means that if the signal strength in a given direction is low, positions close by will also, in all likelihood, also have a low signal strength. Hence it inefficiently tests points that have no hope of being the solution.

Algorithms for finding a good solution faster than an exhaustive search are a major area of research and have been for many years. Genetic algorithms, simulated annealing and other similar algorithms have been used to optimise problems in a huge range of fields since they were first proposed. The literature surrounding these algorithms was studied to find those that may be suited to this scenario (as not all algorithms are suitable for all applications).

Once several suitable algorithms had been found, they were implemented. They were tested for their speed in finding a solution and their ability to find the correct answer. The results are presented across Sections 5.2.1 and 5.2.2, and improvements are suggested.

5.1 Literature review

The purpose of an optimisation is to find the lowest cost solution in a search space. The 'search space' is an n-dimensional space, where n is the number of variables that are to be tuned during the optimisation process. Hence the search space covers every possible permutation of the n variables.

Every permutation of tunable variables has an associated 'cost'. This cost is a mathematical indication of the 'goodness' of a solution. Generally the best position is the one with the lowest cost. An optimisation algorithm is a method of finding a good solution without testing every single possible position in the search space. The tradeoff is that no optimisation method is guaranteed to find the **best** solution; simply a good solution.

The cost that is being optimised can in some cases be expressed as a mathematical function with weights to rank the importance of various factors. An example cost function for this research is shown in Fig. 5.1. It is the result of simulating $\Sigma - \Delta$ for a two element

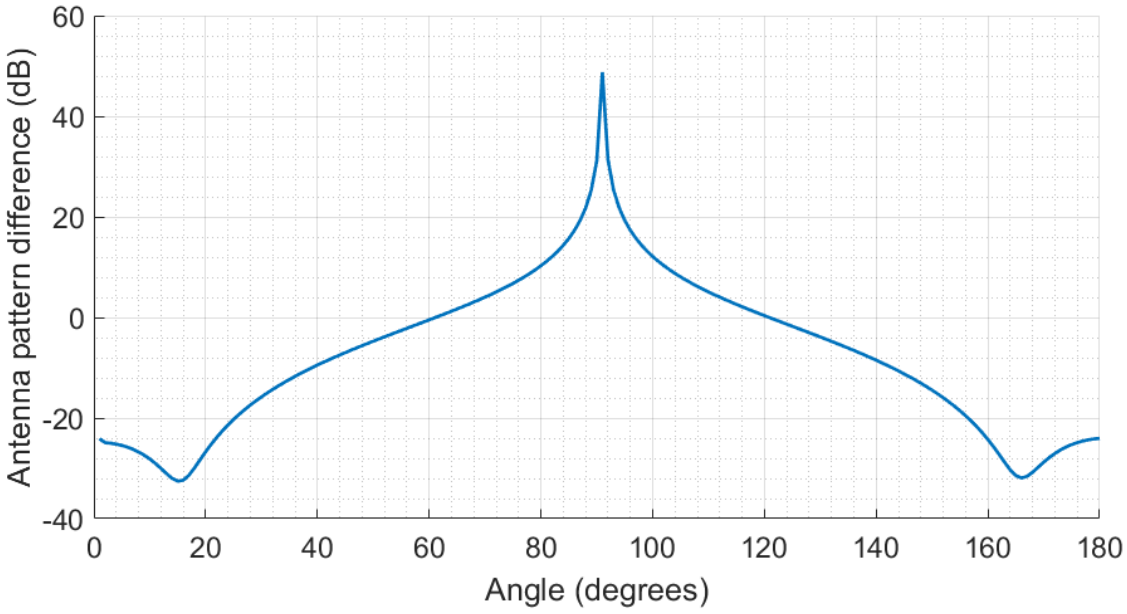


Figure 5.1: Simulated output of $\Sigma - \Delta$ for a two element array.

array, assuming the jammer is broadside to the array. The target is the peak at 90°. For this reason the 'cost' function can be more accurately referred to as an *objective* function, as the optimisation process could be attempting to find a maximum or a minimum, depending on the scenario.

According to Pham [106], optimisation algorithms can be broadly separated into two groups (both of which contain many subtypes): gradient-based methods, and direct search methods. The classification depends on if the algorithm uses derivative information (gradient methods) or not (direct search).

Gradient methods will differentiate the objective (or cost) function, using slope information to find a suitable direction in which to search. This information about the slope is the 'derivative information' as it is found by deriving the cost function. For instance, using the cost function in Fig. 5.1, the derivative would show that if the slope is highly positive or negative, the solution is nearby, and the optimum is found when the derivative goes to zero. However, finding only the zeroes of the derivative function would reveal three potential solutions - one maximum and two minima. The algorithm would have to consider the gradients either side of the zero to determine if it is a peak or a trough.

Gradient methods are unsuitable for this work as they require prior knowledge of the objective function. This prior knowledge is not available because while the impact of a single jammer on the array is known, it varies depending on the angle of arrival, which would move the zero crossing for which the algorithm is searching. The exact incident signal is also affected by scattering, multipath, and noise from the environment. There may also be multiple jammers present, which would confound the system. Aside from not knowing the shape of the objective function in advance, gradient methods often struggle if the objective function has local minima, which is likely in this situation. In summary, gradient methods are not suitable for this work and will not be discussed further.

The second type of search method is direct search. These algorithms require no prior knowledge of the objective function, and the algorithm searches the solution space using

a defined search technique. The value of the objective function is evaluated at each point selected for testing. For instance, in this scenario a beam angle would be chosen and the received power in Σ and Δ measured. The result of the evaluation, and this information alone, is used to determine the search direction for future iterations. The search algorithms to be investigated in this Thesis fall in the direct search category.

Having chosen the broad category of algorithms that will be used, the next step is to look at specific examples. All of these methods are designed to be more efficient than an exhaustive search, but 'efficient' is a somewhat nebulous term which depends on the context. In this Thesis, an efficient method will test as few positions as possible and keep computational complexity to a minimum, as these factors keep the time taken and power consumed to a minimum. With this aim in mind, a number of different algorithms from the literature have been studied to find those worth pursuing.

5.1.1 Genetic Algorithms

The most well-known optimisation algorithm is a Genetic Algorithm (GA). The theory is based upon evolution in nature [107]. The variables to be optimised are imagined to be on 'chromosomes'. An initial population is generated and each chromosome has its fitness tested - that is, the objective function is evaluated for the values in the chromosome. The next step has the population ranked in order of fitness. The most fit solutions of each generation are combined into pairs and allowed to 'breed'. The breeding process takes place as follows: chromosomes from the two parents are aligned. At a randomly generated point the 'genes' (values for individual variables) are switched between chromosomes, producing offspring that are different to either of the parents. In addition to this combination process there is also mutation, which produces new values that did not exist in the chromosomes of either parent. The mutation rate is often relatively low; it varies between implementations, but 4% was used by Boeringer *et al.* [107].

The main reason a GA is not suitable for this problem is the scale of the algorithm. A GA is designed to optimise very large problems; typical algorithms may iterate over 100,000 or more steps before a solution is found. This is excessive in this system, which only has a few variables to optimise. It is known to have low computational efficiency compared to other algorithms [108].

5.1.2 Simulated Annealing

Another well-known optimisation algorithm is Simulated Annealing (SA). This algorithm is not based upon a biological process, but instead the movement of atoms in a cooling crystal lattice structure. Annealing is the process of heating and cooling a material in a particular way to improve toughness. It is well known that rapidly cooling a material will cause it to have a chaotic structure, while cooling it more slowly will allow the atoms to move around and settle into lower energy positions, making the overall object less brittle. If the objective function is the multidimensional lattice structure and the cost function is the energy state of an atom in a given position, this idea can be used to optimise systems.

Simulated annealing is a Monte Carlo method. Monte Carlo methods rely on generating

random samples many times, in the hope that a solution can be found. They are often used when direct computation is difficult [109].

In SA, the objective function is defined as the internal energy of the system [106]. In real annealing a lower energy state is preferable, similar to how a low cost is targeted in optimisation algorithms. The system is initialised with a number of mobile atoms that are able to move around and reduce the internal energy.

Initially the 'temperature' of the system is high. A higher temperature means the atoms have more energy and are able to move further, and potentially occupy higher energy states. As the algorithm progresses the temperature decreases.

With each iteration a new position for each atom is randomly generated. This new position has its energy level tested and compared to the previous position, and the algorithm will decide if the new state is accepted or if the atom remains in the previous state. The probability of the new state being accepted depends on the relative energy levels of the two states, the temperature of the system, and the settings of the algorithm. If the new energy level is lower than the old one, the new position is more likely to be accepted. In some implementations, but not all, lower energy states are automatically accepted. The temperature of the system also influences the probability that it is accepted. When the temperature is higher, the new position may be accepted even if the energy level is higher (*i.e.* worse).

Simulated Annealing, like Genetic Algorithms, is useful when optimising large, multivariate problems. It has the benefit of being able to find a good solution in seemingly random objective functions. However, it is not suitable in this scenario which is small, and would benefit from an algorithm that can make use of knowledge gained as it operates.

5.1.3 Bio-inspired algorithms

Millenia of evolution mean that many biological systems have developed methods of finding solutions for problems. The solution produced by evolution is almost never perfect (see, for example, the blind spot in the human eye), but it is a solution that works. Many researchers have looked to nature to find efficient ways to find good (but not necessarily the best) solutions to problems.

Most bio-inspired algorithms follow the same course, beginning with an initial population that is spread over the search space. With each iteration the positions will focus in on areas that show more promise and neglect areas that do not appear to contain a good solution. The two stages can be referred to as exploration and exploitation [110]. Initially the algorithm will *explore* the search space to gain an understanding of the rough shape. As the operation progresses the population will focus on those areas that show promise. By putting many points in a smaller area, the search becomes finer, making sure full use is made of the area of good potential, thus *exploiting* it. The key difference between each algorithm is the exact method by which the population moves from exploring an area to focusing in and exploiting a few small portions of the search space.

There is a huge range of bio-inspired algorithms that have been proposed. The most well known is Particle Swarm Optimisation, but there are myriad others such as Shuffled Frog Leaping, Invasive Weed Optimisation, Chicken Swarm Optimisation, Paddy Field Op-

timisation and so on. Two algorithms have been chosen for further study: Particle Swarm Optimisation and Invasive Weed Optimisation. Particle Swarm Optimisation was chosen as it appears to be the most well known and so is often used as a benchmark in the literature. Secondly, Invasive Weed Optimisation was chosen as the implementation appears simple.

5.1.3.1 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was first proposed by Kennedy and Eberhart [111]. It is based on how bees will explore a field of flowers. The bees will spread out through an area of flowers and search for pollen. A bee that finds an area with a good food supply will report it back to the other members of the hive. Other bees will then also investigate this area, but they also may remember their own discovery of a promising area. As the bees explore the areas with more food, they will find the local best - after all, the first bee in an area may not have found the very best flower, simply a good one. With time, the bees will find an area with a good food supply but without checking every flower in the area. In the same way, PSO allows a group of particles to share reports of areas with promise (areas with a low cost) so that the search can be focused.

PSO is a computational algorithm that uses a number of potential solutions (referred to as 'particles') which are able to move around the solution space to locate a good solution. The velocity of the particle is randomly generated both initially and during the execution of the algorithm, but each particle experiences a 'pull' towards local and global bests.

PSO is not a Monte Carlo method in the same way that Simulated Annealing is. During the process of SA the new positions to be tested are generated completely randomly. In contrast the particle positions (which result from the velocity of the particles) have an element of randomness but are influenced by information gathered by the particles.

The algorithm works as follows (adapted from [108]), with three main steps after initialisation:

1. (Initialisation) Generate positions and velocities for each particle.
2. Calculate cost of each particle.
3. Calculate the new velocities for each particle.
4. Update positions of each particle; go to 2.

Step 1 is the initialisation of the algorithm. For every particle, a random starting position and velocity are generated. The starting positions are uniformly distributed over the entire search space. The position has as many dimensions as there are variables to optimise, with each taking any value between the minimum and maximum value for that variable. The velocities are equally multidimensional. The velocity is capped so that particles do not attempt to move beyond the search space.

For Step 2, once every particle in the population has a position, their fitness is evaluated. This value is saved as the particle's current value but also the 'local best' (as it is the first iteration). The best value found by any particle is saved as the 'global best'.

After evaluating the costs, the velocity of each particle is updated in Step 3. The velocity is influenced by the particle's current motion, its local best, and the global best, as shown in (5.1). The velocity calculation is simply an expanded version of the equation Velocity = Distance/Time.

$$v_{k+1}^i = wv_k^i + c_1\text{rand}\frac{(p^i - x_k^i)}{\Delta T} + c_2\text{rand}\frac{(p_k^g - x_k^i)}{\Delta T} \tag{5.1}$$

In this equation, v_k^i refers to the velocity of particle i during iteration k, while v_{k+1}^i is the same for the following iteration. The velocity is influenced by three factors: the current motion, the particle memory, and the swarm memory. The current motion is simply multiplied by a weight, w. w is also called the inertia factor, as it is effectively the mass of the particle and therefore its reluctance to deviate from its current trajectory. The particle memory is calculated using the best position encountered by that particle so far, p^i. The larger the distance between that position and the current position of the particle (x_k^i), the larger influence this part has. A random distance between 0 and the distance between the positions is generated, then divided by ΔT to generate a velocity. This velocity can be in any direction. It is then multiplied by c_1, the weight for the particle memory (also known as the self-confidence factor). The final part of the equation considers the best position found by any member of the swarm, p_k^g. As before, a random velocity is generated, with its maximum size dictated by the distance between the current position and the global best position. This is multiplied by the swarm confidence factor, c_2.

Once the new velocity has been calculated, for Step 4 the position x_{k+1}^i is updated using (5.2).

$$x_{k+1}^i = x_k^i + v_{k+1}^i\Delta T \tag{5.2}$$

If the new position is outside the search space, it is moved to be on the edge of the search space and the velocity is reset to zero. Once the positions and velocities have been updated, new fitness values for each particle are calculated. The particle best and swarm best will then be updated as necessary. The algorithm then returns to the velocity calculating stage and this process is repeated until a solution has been found.

In the literature there are examples of the PSO algorithm being used to create beam patterns. For example, Haupt *et al.* use PSO to tweak the phase shifts of each element of a phased array [34]. The goal was to maintain a strong main lobe pointing in the direction of interest (perhaps towards a transmitter) but place nulls in the direction of any noise signals. This is achieved by only allowing the particle swarm to optimise the least significant bits of the digital phase setting. The cost function applied was the overall incident power into the receiver. As the most significant bits could not be changed the optimisation process had little effect on the shape of the main beam, but it was able to alter the side lobes so that nulls pointed in the direction of interferers. This situation differs from the research presented here because the main beam is fixed pointed in a known direction. In contrast, in this research the direction in which the main beam should point is unknown.

The main disadvantage of PSO is its sensitivity to initialisation, meaning if the initial parameters (such as positions, velocities, and the values of w, c_1 and c_2) are not chosen

correctly the algorithm may produce a poor solution. Initialisation of those parameters is entirely dependent on the scenario and so requires experimentation to get good results; this could take a lot of time.

5.1.3.2 Invasive Weed Optimisation

Invasive Weed Optimisation (IWO) was first proposed by Mehrabian *et. al* in 2006 [112]. It is an algorithm superficially similar to Particle Swarm Optimisation, in that a number of particles (or in this case, weeds) are spread around the search space, their costs are identified and the positions of the weeds are updated to reflect better solutions in the next generation. This algorithm was developed to overcome PSO's sensitivity to initialisation parameters.

The IWO algorithm is based on the behaviour of weeds as they adapt to growing in a hostile environment. Weeds will sprout in random locations throughout the environment. Some will encounter more favourable growing conditions than others. Those in better positions are able to produce more, healthier seeds and so have more offspring. A weed is typically a plant that grows quickly, has many offspring and dies easily. They also spread their seeds over a wide area as this offers the greatest chance of survival. This sort of growth characterises the exploration phase of this algorithm's operation. At the other end of the scale are plants such as trees, which grow slowly and produce few offspring. They tend to spread their seeds over a smaller area. The main cause of death for this type of organism is too much competition.

In the same way, the IWO algorithm has a population that initially acts like weeds. Seeds are spread widely, allowing a wider area to be covered. However, as better niches are found, more tree-like growth is possible. At this point competition becomes significant and the weakest individuals are killed off.

The algorithm executes as follows (adapted from [113]):

1. Initialise a number of weeds.
2. Weeds grow and produce seeds.
3. Seeds are spread over the search space.
4. Process repeats from step 2, with culling as necessary.

The first step is initialising the population. The first weeds are spread randomly over the entire search space using a uniform distribution. As with PSO, the multidimensional position is a combination of all of the variables that are to be optimised.

The second step is allowing the weeds to 'grow'. In this stage the cost of each weed's position is calculated. An area that has a good fitness according to the objective function will produce a strong plant, while weeds growing in poorer fitness areas will be weaker. Stronger plants are able to produce more seeds than weaker plants. Once all the costs have been evaluated, the number of seeds assigned to each plant is calculated.

To calculate the number of seeds per weed, first the parent weeds are sorted in order of cost. The parent weed with the best cost is assigned the maximum number of seeds, and the parent weed with the worst cost assigned the least. The remaining parent weeds are arranged linearly between these two points. The number of seeds assigned is then calculated as per

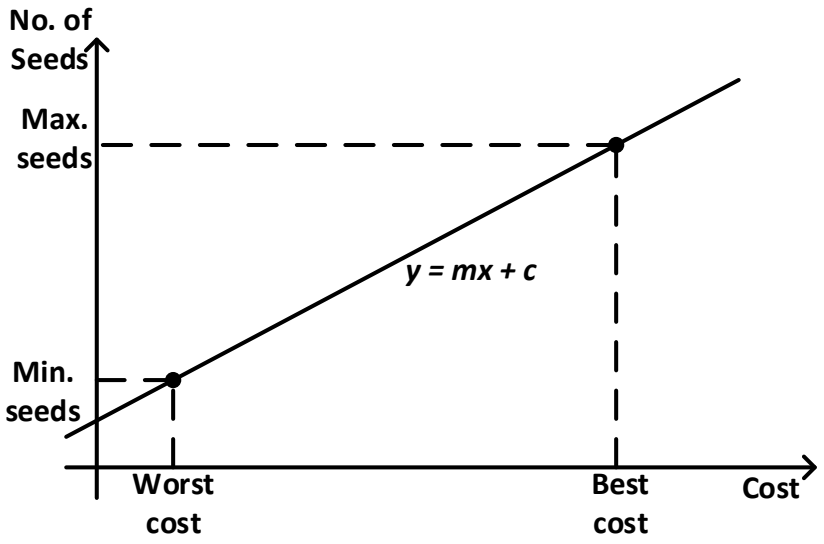


Figure 5.2: Graph demonstrating how the number of seeds assigned to a parent weed can be calculated based on the cost function.

Fig. 5.2. All of the weeds are arranged on the x axis and the line between the best and worst parents (labelled $y = mx + c$) allows the number of seeds to be read.

Once the number of seeds has been allocated, new weeds are generated. The seeds of the fittest parent are the first to be assigned, with the algorithm moving through the parent weeds from best to worst until the number of plants (including the new seeds) meets the maximum population. In this way the strongest weeds with the highest fitness reproduce the most, allowing the algorithm to test areas that show more promise. However, lower ranked weeds are still able to reproduce, reducing the chance of the algorithm becoming trapped in a local minimum.

The third step assigns the positions of the seeds. The seeds are placed randomly, according to a normal (Gaussian) distribution, within a certain range from their parent plant. The allowed range varies between iterations according to 5.3,

$$\sigma_m = \frac{(m_{\max} - m)^n}{(m_{\max})^n}(\sigma_i - \sigma_f) + \sigma_f \tag{5.3}$$

where σ is the standard deviation of the spread for a given iteration, m is the current iteration number and $m_{\max}$ is the maximum number of iterations, i is the initial value of σ for the first iteration, and σ_{f} is the deviation for the final iteration. n is the modulation index, which controls the rate at which the range changes.

Early on in the process, the spread is large. This allows the weeds to be scattered widely and more of the solution space is explored. As the number of iterations increases, the range narrows and the search focuses in on the areas with the most potential. This change in range controls the mode of the algorithm as it moves from exploration to exploitation; it explores when the range is large, and exploits when the range is small.

The final step of the algorithm takes into account overpopulation. The processes of growth and reproduction will continue until the maximum population is attained. Once the number of weeds exceeds the maximum that can be supported in an area, the weaker plants will die off. To achieve this, all weeds are ranked from most to least fit. Then, all the weakest plants

past a certain maximum are culled. The algorithm then continues with cycles of growth, reproduction and culling until a solution is found.

5.2 Implementation

The chosen algorithms, PSO and IWO, are both suitable for use in interference detection as they are not gradient methods, meaning they need no prior knowledge of the objective function. They also have relatively low computational complexity (compared to, say, GAs and SA). This reduces the power consumption and cost of the overall system.

To use these algorithms, the problem needs to be framed in terms that suit the algorithm. In the research for this Thesis the cost (or objective) function is the value of $\Sigma - \Delta$. The algorithm will be attempting to maximise this value as this occurs when the narrow null is pointing at the source of the signal.

The space which is being searched is the range of values the phase shifters can take. In the case of a two element, one dimensional antenna (which is being considered from now on) there are two phase shifters that can be changed. There are no other components (at this point) that need optimising. As the beam angle is actually a function of the *difference* in phase shift applied to the two elements, and not the absolute phase shift of each element, the problem can be reduced to only having one dimension by fixing the phase shift applied to one element. Hence the problem only contains one variable, making the search space one dimensional. In addition, the possible values of the phase shifter are integers only, so the search space is simply the integers between 0 and 255 inclusive. The algorithm will vary the phase shift to one element of the array in an attempt to maximise the output of $\Sigma - \Delta$.

There are two main methods for deciding if a solution has been found [114]. In both cases, a maximum number of iterations will have been defined. The first option for finishing the algorithm's execution is allowing the algorithm to run until the maximum number of iterations has been reached. This method can be inefficient as the maximum number of iterations needs to be sufficiently large to reliably find a good result. However, on some occasions a solution will be found much faster and so time will be wasted waiting for the maximum number of iterations to be reached without further improving on the solution. The other method for ending the program is to detect when the best solution found is not improving between iterations - when a plateau has been reached [115]. This method can produce a solution more quickly as it does not always take as many iterations. However, it becomes more likely that the best solution will be missed as often the algorithm will pass through a number of plateaux, improving the answer in steps. Plateau detection may finish prematurely; however, its fast operation means this is the method that is used in this research.

Other methods include finishing the process if all the particles are within a certain area, arguing that if they have all congregated, they are likely gathering near the best solution and not just a local solution. However, this method will not be considered because the cost function is highly discretised (integers only) and the search space relatively small (only 256 points), meaning the particles may end up all testing the same position (which is inefficient) if the area is set small and the algorithm may terminate prematurely if the area is larger.

5.2.1 Particle Swarm Optimisation

A version of the PSO algorithm described in Section 5.1.3.1 was implemented. The full code is found in Appendix A. Key parts of the program's operation are highlighted. All line numbers refer to Appendix A.

The program is written in Python. This is an object-oriented language, meaning instances of 'objects' can be created. Each object has a number of variables associated with it. In this program, each particle is programmed as an object (lines 24-33), and the variables associated include the phase shifter setting, the velocity, and the position and cost of the local best. The particle objects are stored in a list for easy manipulation.

The program uses `sys.argv` to check for command line arguments, lines 35-41. If, when the program is run, arguments are added, this part of the code will identify them. This is important as it allows variables to be changed from outside the program; that is, a script can be run that will test multiple different permutations of variables without manual intervention.

This implementation of the PSO algorithm is for simulations. To make the simulations realistic, a sweep recorded at Sennybridge is used as the objective function. When the system evaluates the cost function, it is in fact reading the value of the recorded sweep at the phase setting chosen by the PSO algorithm.

First, the existing particles are sorted according to cost. This process uses Python's built in `sort()` function, which uses a type of merge sort algorithm [116]. Tests have shown it to be fast and have low memory usage, although this was not verified for this work. As this is a proof of concept system it was accepted that the sort must only be functional, and it can be optimised later as necessary. Once the data have been sorted, a shift register is used to store the previous three generations' best results. It is then used to decide if the best cost has been improved upon, or if a solution has been found.

Lines 119 to 142 calculate the new velocity for each particle, according to (5.1). The three terms (inertia, particle memory and swarm memory) are calculated individually and added together. The velocity is also capped to a maximum of 256, as there are 256 possible positions. The position is then updated using this new velocity. The original algorithm declared that if the new position was outside the search space the position should be moved to the edge of the search space, and the velocity should be set to zero. This was modified for this implementation. If the particle went beyond the search space, the excess travel was calculated and subtracted from the opposite side of the search space to find the position, making the search space effectively circular. This decision was made because the real system has the same circular nature and a phase shift of 358° is 2° away from a phase shift of 0°. Restricting the particles' movement across the 'join' may prevent a correct answer from being found.

Tests with the IWO algorithm found that if the answer had improved by less than 5% over three generations, a plateau could be declared reached. This is discussed in more depth in Section 5.3.

As this particular implementation of PSO is used for scripting, the final stage is to save the relevant information for later. In this case the important information is the best position found by the swarm and the number of positions tested. In addition the values of w, c_1 and

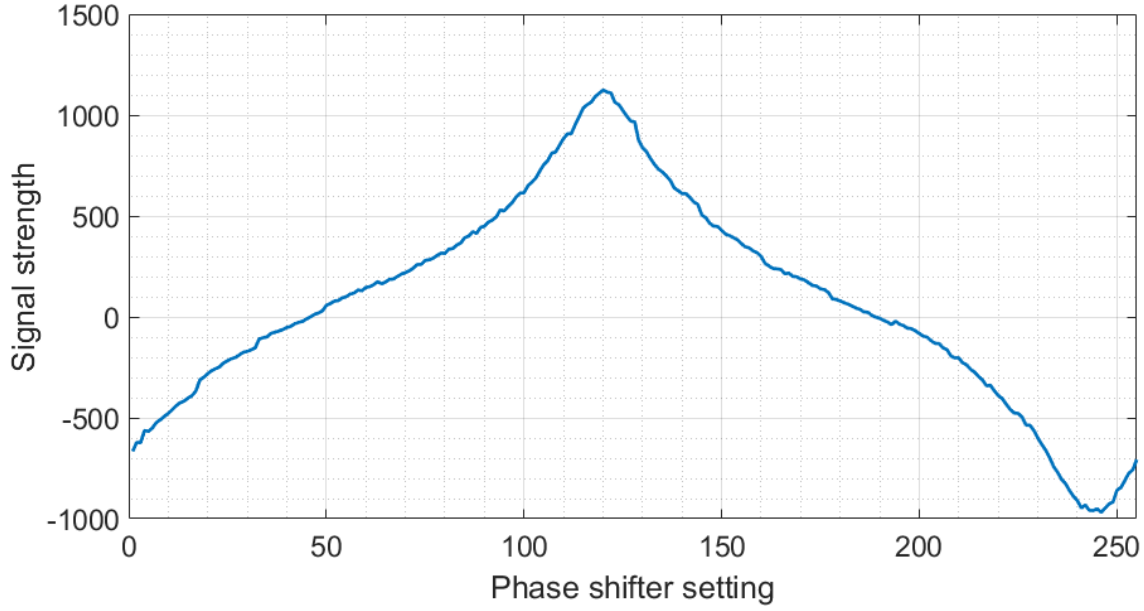


Figure 5.3: The objective function used in the simulations for in this Chapter.

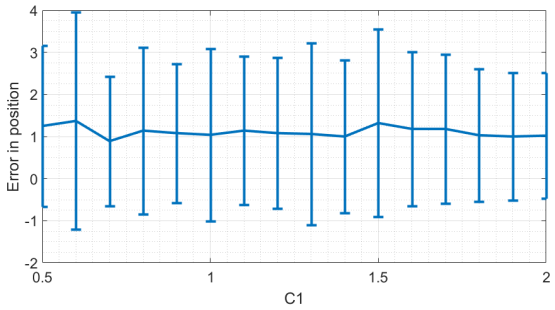
c_2 are saved, as the combinations of these variables are being tested. The 'call number' is for debugging only - it is a count of how many runs of the same combination of w, c_1 and c_2 have taken place.

5.2.1.1 One dimensional

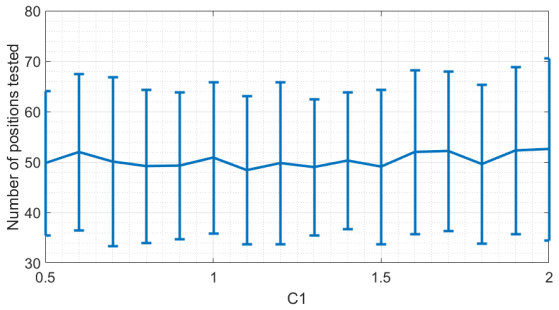
Once the basic PSO algorithm had been written, the next step was to optimise the parameters. The optimum values of w, c_1 and c_2 will vary depending on the exact object function being evaluated and so values cannot necessarily be taken from the literature. The literature suggests ranges of values to use for w, c_1 and c_2. The original algorithm suggested values of 0.4 to 1.4 for w, 1.5 to 2.0 for c_1 and 2.0 to 2.5 for c_2 [111]. Other research found that optimum values might not necessarily be found in this range, setting $w = 0.5$, $c_1 = 1.5$ and $c_2 = 1.5$ [108]. Hence a wide range of values for each parameter is tested in this Thesis.

Figs. 5.4 to 5.6 were produced by stepping one of the three variables through their range of possible values, while holding the other two variables constant at the value in the centre of their range of possible values. Each set of variables was tested over 100 repeats. For each variable being studied, the mean and standard deviation are plotted for the error and the number of positions tested. The error was calculated as the mean of the absolute error, meaning it will always be positive. The standard deviation of the error also uses the absolute value of the error.

The error is not quoted in degrees, but in phase shifter steps. Hence there are 256 possible positions, and the error for each run is an integer value. The error is an important metric as the algorithm must be able to accurately determine the direction of the jamming. According to simulations in Chapter 2, the null of the Δ output is 0.34° wide (measured as 3 dB up from the lowest point). The phase shifters step in intervals of 1.4° (one Least Significant Bit, LSB, of the phase shifter). A phase change of 1.4° moves the beam angle by approximately 0.5°. Hence an error of one LSB would position the solution found outside of the null by this metric. However, it can also be stated that if the error in position is 1° (equivalent to an

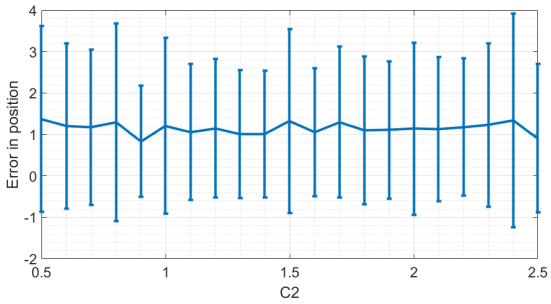


(a) Error in position with changing c_1.

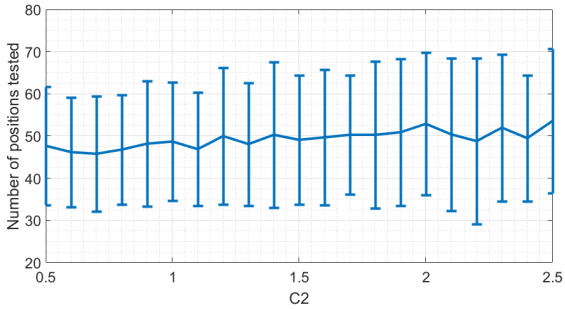


(b) Number of positions tested with changing c_1.

Figure 5.4: Effect of changing c_1, averaged over 1000 runs.



(a) Error in position with changing c_2.



(b) Number of positions tested with changing c_2.

Figure 5.5: Effect of changing c_2, averaged over 1000 runs.

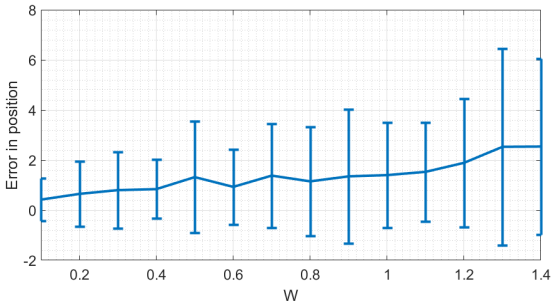
error of two LSBs), the received signal from the jammer is approximately 20 dB lower than it would be in the main lobe of the radiation pattern. At this level of attenuation it could be considered mitigated. Further experimentation would be required to prove it, but for this work an error of two LSBs will be considered to be acceptable.

The number of positions tested is also important as this gives some idea of how long it will take the system to locate the interference source. Assuming that the processing time scales linearly with the number of positions tested, testing fewer positions will find the solution faster.

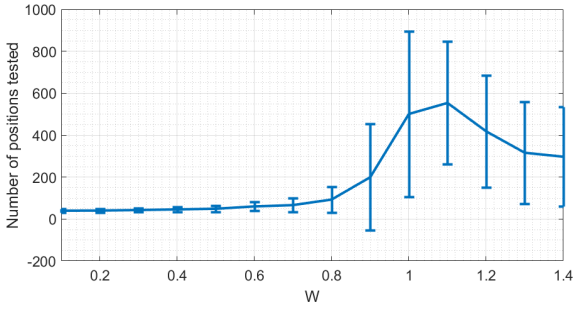
Fig. 5.4 shows the effect of varying c_1 on the number of positions tested and the error. Fig. 5.4a indicates that the self-confidence factor has very little influence on the outcome. The error is consistently around 1°. The standard deviation varies but shows no consistent trend. This may be because with a relatively smooth cost function, consistent improvement means the 'pull' exerted by the local best term is very small, and so the self-confidence factor has almost no effect on the velocity and the overall movement of particles. The same (lack of) trend is observed in the number of positions tested, which remains at around 50 positions.

Fig. 5.5 shows the effect of changing c_2. As with c_1, the effect is small. There is no trend in the overall error, which remains at around one phase shifter LSB on average. Increasing the value of c_2 does slightly increase the number of positions tested, although as before the standard deviation is very large and the confidence in this trend is low. Overall it would appear that varying c_2 has little effect on the error or speed of the algorithm.

Random number generation is required when calculating the velocity of the particles. A random number is generated between zero and the maximum pull towards the local and



(a) Error in position with changing w.



(b) Number of positions tested with changing w.

Figure 5.6: Effect of changing w, averaged over 1000 runs.

global bests is generated. In the implementation given in Appendix A, the built in Python `random.random()` function is used. This function implements a uniform distribution, although it is not guaranteed to be truly uniform. As it has been noted that the values of c_1 and c_2 have no discernible effect on the number of positions tested or the overall error, it is also assumed that the distribution of the random function can also be ignored.

The effect of changing w is much more dramatic. A low value of w generally produces a lower error with a smaller spread. As the value of w increases past one, the error and range increase. Even more noticeable is the effect of changing w on the number of positions tested. When the value of w was above 0.8, the number of positions grew very rapidly. Closer inspection of the data revealed the reason. Once a particle reached a high enough velocity, it would begin to move very quickly across the search space. Large pulls towards the global best, combined with a high level of inertia, rapidly increased the velocity past its maximum. It would be clipped to the maximum velocity in each iteration but the inertia factor, w, would continue to keep the velocity high. The system could be considered to be unstable and oscillating, but with the magnitude of the oscillations being clipped. At this point the algorithm is not working as desired as the particles are moving effectively randomly instead of trending towards a global optimum, and the finding of a good solution becomes almost down to chance.

There is a noticeable decrease in the number of positions tested, after the peak at $w = 1.1$, as w increases further. There is no clear reason for this decrease. Further tests were not conducted at higher values of w as the improvement seemed to plateau between $w = 1.3$ and $w = 1.4$, and the number of positions tested is much higher than for low values of w, so it was deemed unlikely to produce a better solution.

The optimum combination of values, which will be used in further tests in Section 5.2.3, is as follows: $w = 0.4$, $c_1 = 1.5$, and $c_2 = 1.4$.

5.2.2 Invasive Weed Optimisation

The PSO algorithm is the most well known, but a second algorithm was also implemented for comparison. The IWO algorithm was chosen for its relative ease of implementation.

The Python code for the IWO algorithm is given in Appendix B. The PSO code discussed above was designed for simulations. To demonstrate a wider range of code, the version discussed here is capable of interfacing with the hardware designed in Chapter 4. As a result

it is missing some parts, such as reading of cost functions and plotting of results, and has other parts added.

The IWO implementation makes greater use of functions than the PSO algorithm. This reflects its slightly higher computational complexity (although it is still simple compared to algorithms such as GAs and SA), but also the additional computation required to interface with hardware. `ComposeCommand` is a function that accepts a chip type, address and value and creates an eight byte command to be transmitted to the ARM microprocessor. The command structure is identical to that used in the exhaustive sweep searches discussed in Chapter 4. Another function, `CalculateCost`, reads the ADCs and computes the values of Σ and Δ (so that they can be saved for debugging purposes) as well as the value of the objective function. `GenerateWeed` accepts variables indicating the location of the parent weed and the range, and generates a position for a seed.

As this algorithm is interfacing with hardware, it requires the Serial Python library, which allows communication over USB using a serial link. The program also imports a Look Up Table (LUT) for optimising the phase shifter setting.

Lines 155-157 set one phase shifter to a fixed value of 180°. From this point the setting of this phase shifter will not change.

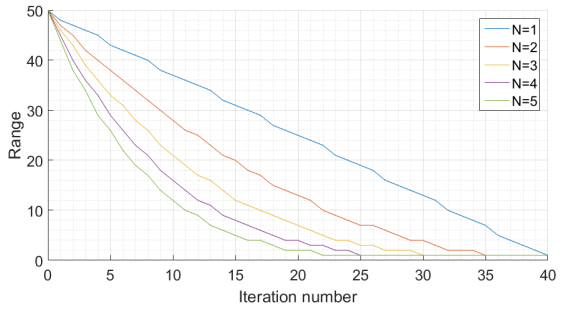
As with PSO, the algorithm proper begins with sorting the population in descending order (this is possible on the first iteration as an initial population will have been generated prior to this point). Having sorted and culled the population, and checked for a solution, the algorithm generates the new seeds. This is a more involved process than updating the velocities of the PSO population. An equation of the form $y = mx + c$ is calculated, so that the number of seeds for each parent weed can be calculated in accordance with Fig. 5.2. This algorithm saves the phase shifter settings and the values of Σ and Δ for every weed so that they can be compared with known objective functions. This is discussed in more detail in Section 5.3.

Before the algorithm's performance can be compared to that of PSO, it must also be optimised. The key to this algorithm's performance is the calculation of the range (the distance over which seeds can spread from their parent weed). The equation governing this range is given again here for reference.

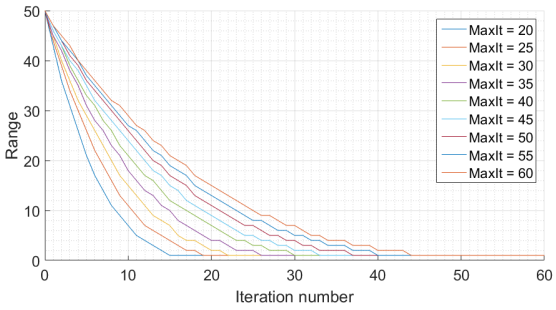
$$\sigma_m = \frac{(m_{\max} - m)^n}{(m_{\max})^n}(\sigma_\mathrm{i} - \sigma_\mathrm{f}) + \sigma_\mathrm{f} \tag{5.4}$$

The values of σ_i and σ_f, the initial and final ranges, are held constant. They are chosen so that in the first iteration the entire search space is available for weeds to be planted in, while in the final iteration the range is the smallest possible (*i.e.* one). The original algorithm used a normal distribution so to get good coverage over the whole search space, the value of σ_i was set so that the edge of the search space was three standard deviations away from the centre. For these illustrative graphs, the value of σ_f has been rounded to 50.

The IWO algorithm has a term named the nonlinear modulation index, n. It affects how the range changes between iterations. A low value of n causes a gradual reduction in the range, while higher values cause the range to drop quickly at first, and more slowly towards the end. Controlling the rate of change of range controls the behaviour of the algorithm, as

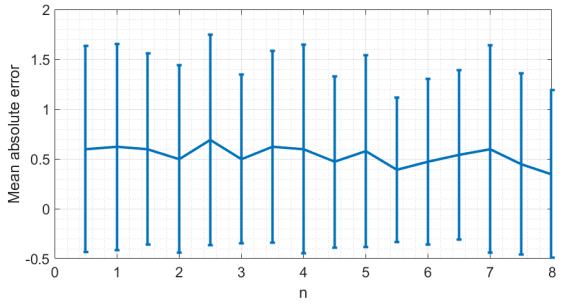


(a) Effect on the range with changing n.

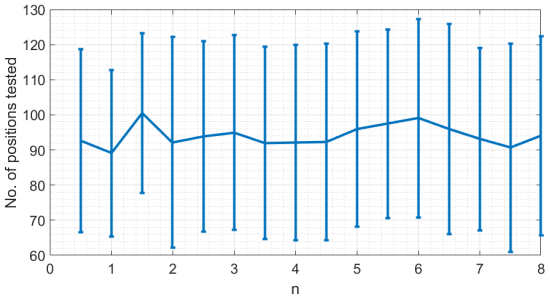


(b) Result of changing the maximum number of iterations, fixing $n = 3$. σ_i and σ_f fixed.

Figure 5.7: Effect of changing variables on the range over the course of the algorithm's run. Range values are rounded to the nearest integer.



(a) Error in position with changing n.



(b) Number of positions tested with changing n.

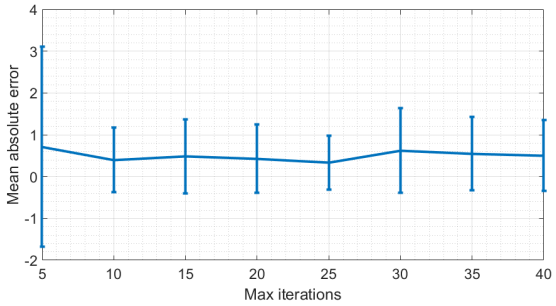
Figure 5.8: Effect of changing n on the error and number of positions tested, averaged over 1000 runs.

with this algorithm the exploration and exploitation phases of operation are closely tied to the range. The effect on the range of changing n is shown in Fig. 5.7a. The start and end points are fixed by σ_i and σ_f.

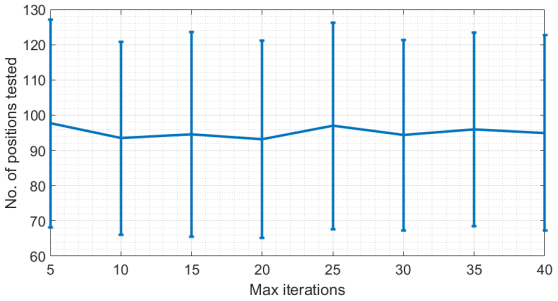
Experimentation in [112] showed that the optimum value for n was three, therefore this value was used henceforth. Fig. 5.7b shows the effect of changing the maximum number of iterations while n is held constant.

While the literature had found that the best results were obtained by setting $n = 3$, tests were still run to verify that this value was suitable. A number of values for n were tested, from 0.5 to 8, with 100 repeats of every test. The value of σ_i was 50 and the maximum number of iterations was 50 throughout the tests. The results are shown in Fig. 5.8. Fig. 5.8a shows a slight downward trend in the error as the value of n increases. The number of positions tested (Fig. 5.8b) shows no overall trend. Hence there may be a small benefit to increasing the value of n, but the data are not conclusive. The standard deviation for all values of n is very large, so the mean values given should not necessarily be trusted.

Fig 5.9 shows what happens when the value of the Max Iterations is varied between 5 and 40 in steps of 5, while $n = 3$ and $\sigma_\mathrm{i} = 50$. As when n was varied, there is little variation in the mean error or the number of positions tested. There is one exception: if the maximum number of iterations is small (5) the error is slightly higher and much more variable, but the standard deviation is much higher. This is probably because the error after five iterations becomes much more dependent on the initial distribution of weeds: if one is near to the final

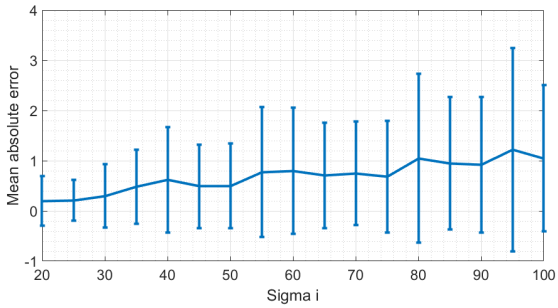


(a) Error with changing the maximum number of iterations.

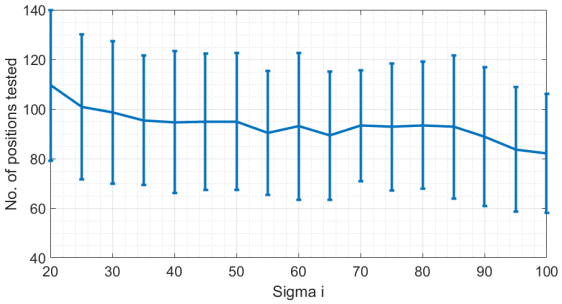


(b) Change in number of positions tested with changing the maximum number of iterations.

Figure 5.9: Effect of changing the maximum number of iterations, averaged over 1000 runs.



(a) Error with changing σ_i.



(b) Change in number of positions tested with changing σ_i.

Figure 5.10: Effect of changing σ_i, averaged over 1000 runs.

answer the error will be low but if there happens to be no initial positions near the solution, after just five iterations it may not have been found. When more iterations are used, the algorithm becomes insensitive to the initialisation conditions.

Fig. 5.10 shows the effect of changing the value of σ_{i}, which controls the maximum range over which the weeds can spread at the beginning of the process. A small value of σ_{i} results in a very small error but a relatively high number of positions tested. Conversely a larger value of σ_{i} results in a larger error but a small number of positions tested (on average 30 fewer when $\sigma_{\mathrm{i}} = 100$ compared to when $\sigma_{\mathrm{i}} = 20$). Hence a middling value is chosen, to get a balance of the low error and low number of positions tested, and so $\sigma_{\mathrm{i}} = 40$. The larger error with a higher value of σ_{i} may be due to the way it increases the initial range. If the range is high, a good position may not be improved upon within three iterations (the number of iterations over which an average is taken for determining plateaux). Hence the algorithm will declare that a solution has been found even if the solution is not the best possible. This would also explain the increase in the standard deviation in the error with higher values of σ_{i}. The decrease in the number of positions may be due to the fact with a wider range, if there is gradual improvement and the algorithm does not terminate prematurely, it is able to make larger steps towards the solution than if the range is kept small by a small initial σ_{i}.

Like the PSO algorithm, random numbers are generated during the execution of the IWO algorithm. Unlike PSO, the random number generation plays a much bigger part, as it directly determines the positions of the seeds, as opposed to having some unquantified effect on the velocity. Hence experiments were run on the effect of using a normal or a

Table 5.1: Effect of changing distribution type on error, number of positions tested and number of iterations when using IWO. Means given, standard deviation in brackets.

	Normal distribution	Uniform distribution
Error in position	0.23 (0.50)	0.64 (0.85)
No. of weeds	85.7 (22.3)	92.8 (26.2)
No. of iterations	3.36 (0.51)	3.70 (0.63)

uniform distribution. Table 5.1 gives the averages over 1000 repeats. The mean value is given, with the standard deviation of the results in brackets. The normal distribution, which is recommended by the literature, performs better overall than the uniform distribution. The mean error is smaller for the normal distribution (0.23 compared to 0.64), meaning it finds the correct solution more often. The number of positions tested ('No. of weeds') is also smaller (85.7 compared to 92.8), meaning the algorithm should take less time to run. On average it also uses slightly fewer iterations to find a solution (3.26 compared to 3.70). The number of iterations is important as there is some computational overhead associated with each iteration (sorting the weeds, calculating the number of seeds per weed and so on), so fewer iterations is preferable. The exact reason that the normal distribution produces better results is not clear, but it may be to do with the fact it biases the seeds towards the parent weed, and has a similar effect to reducing the range.

To get the best results, the parameters for IWO were set as follows: a normal distribution was used, and $n = 3$. Both of these decisions were recommended by the literature and backed up by experiments. The other two values, σ_i and the maximum number of iterations, were chosen purely experimentally. The best results were obtained when $\sigma_\mathrm{i} = 50$ (chosen to balance between errors and the number of positions tested) while the maximum number of iterations was 40.

5.2.3 Comparison of Particle Swarm and Invasive Weed Optimisation in simulations

The comparison in this Section was accepted for presentation at the (cancelled) European Conference on Antennas and Propagation (EuCAP) in March 2020 [46].

Having optimised the parameters for each algorithm, the next stage was to compare the performance of the two algorithms. To have a good comparison, a number of objective functions were generated. Each objective function had the solution at a different point, which would simulate the jammer being present at a different angle. The positions ranged from 10° to 90° in steps of 10°. Each algorithm was tested 100 times for each solution and an average was taken.

Fig. 5.11 gives the absolute error in position from IWO and PSO. As previously, the error is given in phase shifter LSBs. The standard deviation is also given as error bars. Both algorithms have similar mean errors. Generally the mean and variance are both slightly lower for IWO. This is not the case when the error was close to endfire and the IWO algorithm was used, when the mean number of positions increased greatly and had much greater variance. This is despite the algorithm having 'wrapping', meaning it should not be biased towards the centre of the search space. This may be a fault of the randomly generated numbers, or it

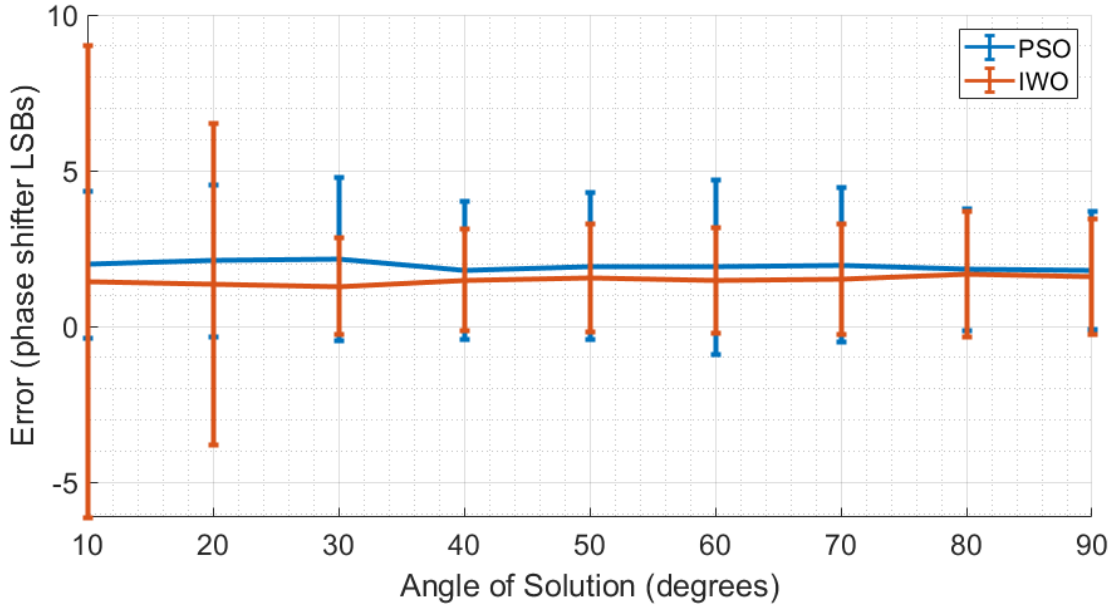


Figure 5.11: Comparison of error achieved by PSO and IWO.

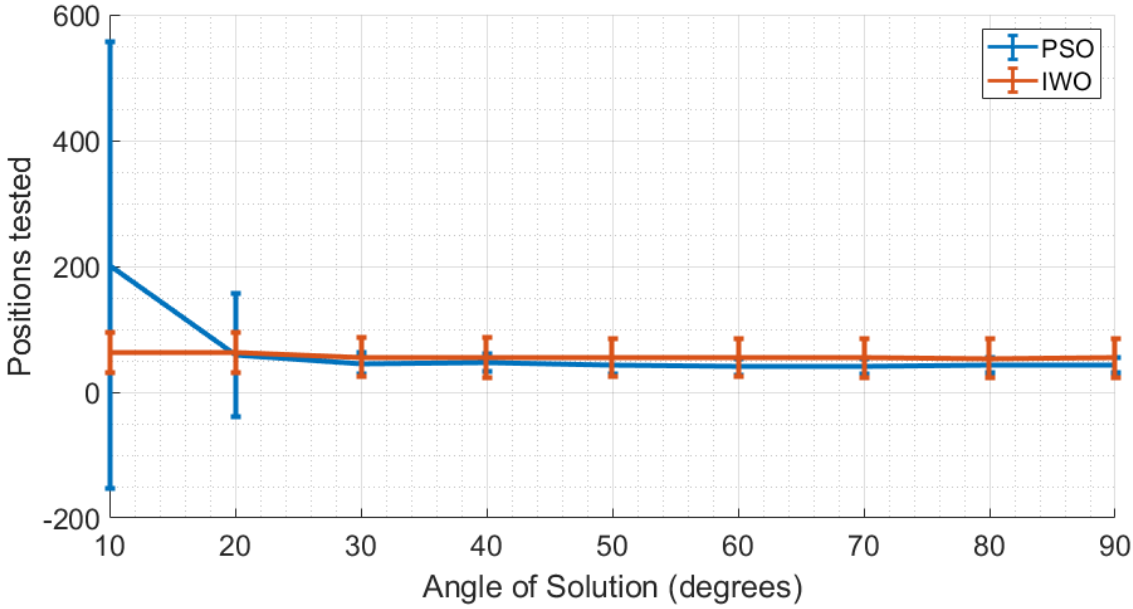


Figure 5.12: Comparison of number of positions tested by PSO and IWO.

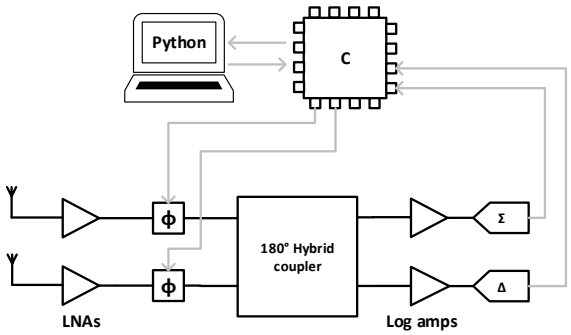


Figure 5.13: Block diagram of the setup for the IWO field tests.

may be something intrinsic to the IWO algorithm.

Fig. 5.12 shows the mean number of positions tested by each algorithm. Generally PSO tests slightly fewer positions and has a slightly smaller variance than the IWO algorithm. When the solution is close to endfire the variance increases significantly. When the solution is at 10° the number of positions increases significantly for PSO. For IWO the average number of positions tested, and the variance in the number, does not change with the angle of the solution.

The general trend appears to be that IWO tests more positions to get a better answer (with a smaller error), while PSO tests fewer positions at the expense of a slightly greater error. This generalisation is inverted at angles close to endfire, where IWO does not test any more positions and has a greater variance in error, while PSO tests many more positions but still achieves a good result without significant variance.

5.3 Field testing of Invasive Weed Optimisation

The performance of the two algorithms is similar, but the performance of PSO appears to be marginally better. However, the decision was made to only implement the IWO algorithm for field tests with real jammers and hardware. The reason for this is because the similar performance meant only implementing one was reasonable, and IWO was chosen as it has not been studied as much in the literature and therefore the research potential was higher. The work described in this Section was presented at the European Conference on Antennas and Propagation (EuCAP) in Krakow in 2019 [45].

The one dimensional algorithm was implemented in Python on a PC. The hardware setup was the same as was used for the exhaustive sweeps. It is shown in Fig. 5.13. The Python interfaced with a microcontroller over USB. The microcontroller changed the phase shifters in accordance with commands from the PC. On-board ADCs were used to read the values of Σ and Δ. The values were then reported back to the PC to be used by the algorithm.

The algorithm was tuned using the values of σ_{i}, n and the maximum number of iterations found during simulations. The simulations are considered realistic as the cost function used was created from data recorded using this hardware in a real environment, using a real jammer. As a result the simulations take into account all of the front-end hardware (not just the antenna, as is the case in simulated beam patterns) as well as the effect of the jammer and the environment.

As before, tests were run at the Sennybridge Training Area in Wales, UK. The tests were designed to demonstrate that the system was capable of locating a jammer under a variety of different conditions; however, they were not exhaustive proof and simply a demonstration. Only one jammer was used but it was held at different distances (varying the signal strength) and at different angles relative to the broadside of the antenna. In all tests, the jammer was placed in its test location and turned on. An exhaustive sweep was run, followed by the IWO algorithm. The sweep allowed the results from IWO to be matched to an accurate picture of the signal strength in each direction. The jammer was not moved or touched between the two experimental runs.

As with the exhaustive sweeps, the system is relatively slow to find jammers. This is due to the communications bottleneck between the PC and the microcontroller. Hence in this Chapter timings will be considered in terms of the number of positions tested and not the absolute speed of the system.

The first test had the jammer placed broadside to the antenna array, approximately 10 m away. The results are shown in Fig. 5.14. The black line shows the results of the sweep, while the points are the positions tested by the IWO algorithm. They are coloured according to which iteration tested that position; the initial population is iteration one. The x-axis values are the angle of the peak of $\Sigma - \Delta$ relative to endfire of the antenna. As with the sweeps in Chapter 4, the value on the y-axis is the ADC counts. These are the values returned by the 12-bit ADC on board the LPC1549 microcontroller. The values are not calibrated but are proportional to the log of the power in the signals.

The correct solution as shown by the sweep was at approximately 94°. The IWO algorithm found a solution at approximately 95.4°. The two tests were run serially and changing scattering and fading could easily be behind such a small change in best position. This is supported by the fact the magnitude of the IWO points are generally different to the sweep at the same angle.

The next tests covered when the jammer was positioned away from broadside. As expected the IWO algorithm locates the correct direction when the deviation was small, so an additional experiment tested if the algorithm would fail if the solution was a long way from broadside. The results are shown in Fig. 5.15. Once again the system is able to locate the jammer, although the IWO points appear to deviate from the results of the sweep. In addition, a higher number of positions (117) were tested. The increase in positions tested may be due to the peak being less sharp. The deviation may be due to the environment changing slightly between runs. This test shows the effect of the algorithm 'wrapping', as many positions are tested when the beam angle is between 0 and 40°, despite these positions appearing almost 360° when depicted on a rectangular graph.

For the tests so far, the tolerance has been set at 5%. The tolerance dictates how flat a plateau must be before the algorithm determines that a solution has been found. For the following tests the threshold was changed to 2%.

Fig. 5.16 gives the result when the jammer is broadside to the array, but further away than the test shown in Fig. 5.14. As a result the peak of the sweep is slightly lower and less sharp. The IWO algorithm does find an answer that agrees with the sweep, but many more positions are tested. In this test, 171 positions were measured before a solution was

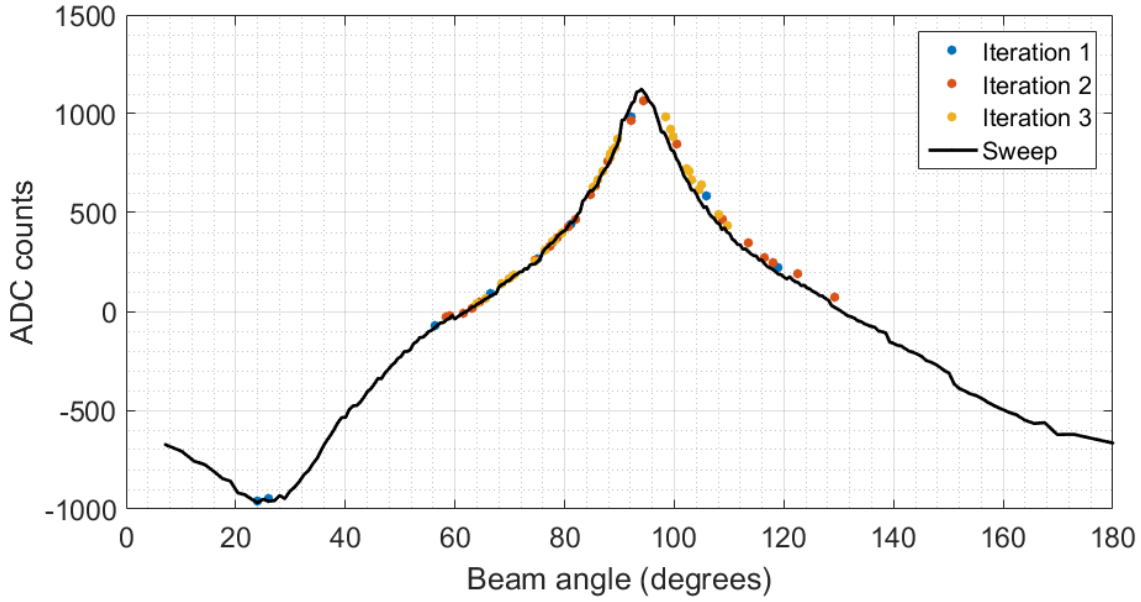


Figure 5.14: Result of an IWO test (points) with the jammer broadside to the array. The tolerance was 5%. A total of 63 positions were tested. An exhaustive sweep of the same scenario is plotted as a line.

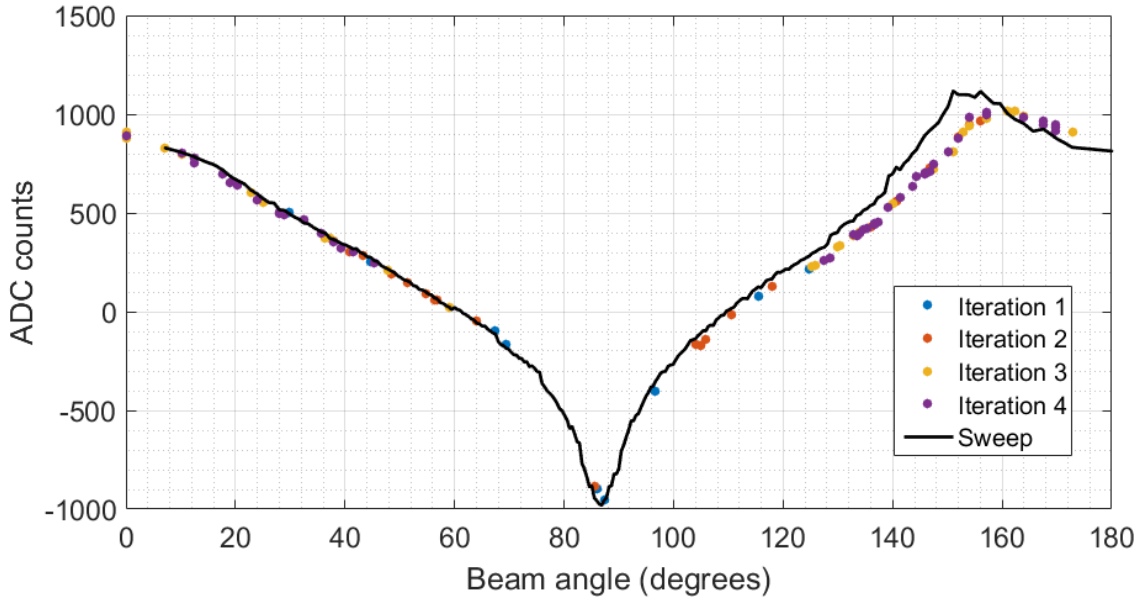


Figure 5.15: Result of an IWO test (points) with the jammer 55° relative to broadside. The tolerance was 5% and 117 positions were tested. An exhaustive sweep of the same scenario is plotted as a line.

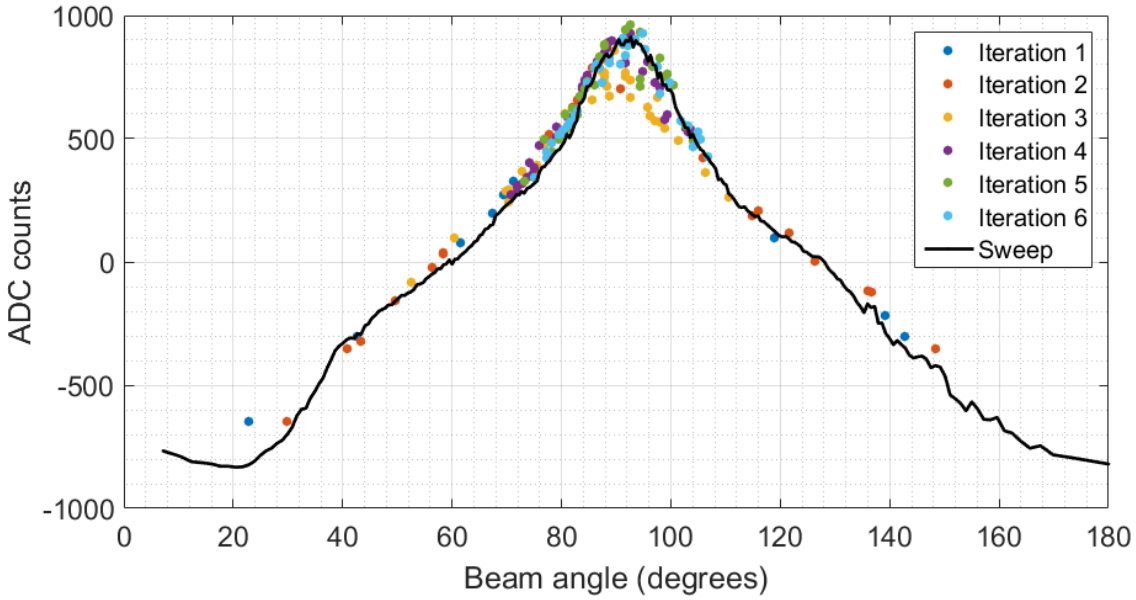


Figure 5.16: Result of an IWO test with the jammer broadside to the array. The tolerance was changed to 2%. 171 positions were tested in total. An exhaustive sweep of the same scenario is plotted as a line.

declared to be found. This is still less than the 256 points required for an exhaustive sweep, but more than is desired. The increase is because the system has become more sensitive to small changes when deciding if a plateau has been reached. During the course of the test (which took a time on the order of several seconds) the scattering in the environment will change slightly, making two points at the same angle have different magnitudes. The higher number of positions tested may also be to do with the less sharp peak making the 'correct' answer less definitive.

It is noticeable that when the peak is not clear and the tolerance is low (2%), the same positions are tested repeatedly. This indicates that the algorithm is not working efficiently and is simply wasting time. As the solution is no better than when the tolerance is set to 5%, it would imply a stricter tolerance does not improve the accuracy but does increase the number of positions tested, meaning it is detrimental to the overall performance.

The final test had the jammer held closer to the antenna. In this case the peak of the sweep signal is sharper and at a higher amplitude (around 1300, compared to around 900 for the previous test). These features made it easier for the algorithm to locate the jammer and only 60 positions were tested. The solution is at 94.4° according to the sweep, and at 95.8° according to the weeds. The difference is 1.4°, which is small enough to be due to noise. Once again the solution is approximately the same accuracy as when the tolerance is 5%, but there will be potentially many more positions tested (although not necessarily). Following these two tests a decision was made to only use a tolerance of 5%.

These tests demonstrate that the algorithm is able to locate a jammer in a realistic environment, even if the jammer is far away from broadside or has a low magnitude.

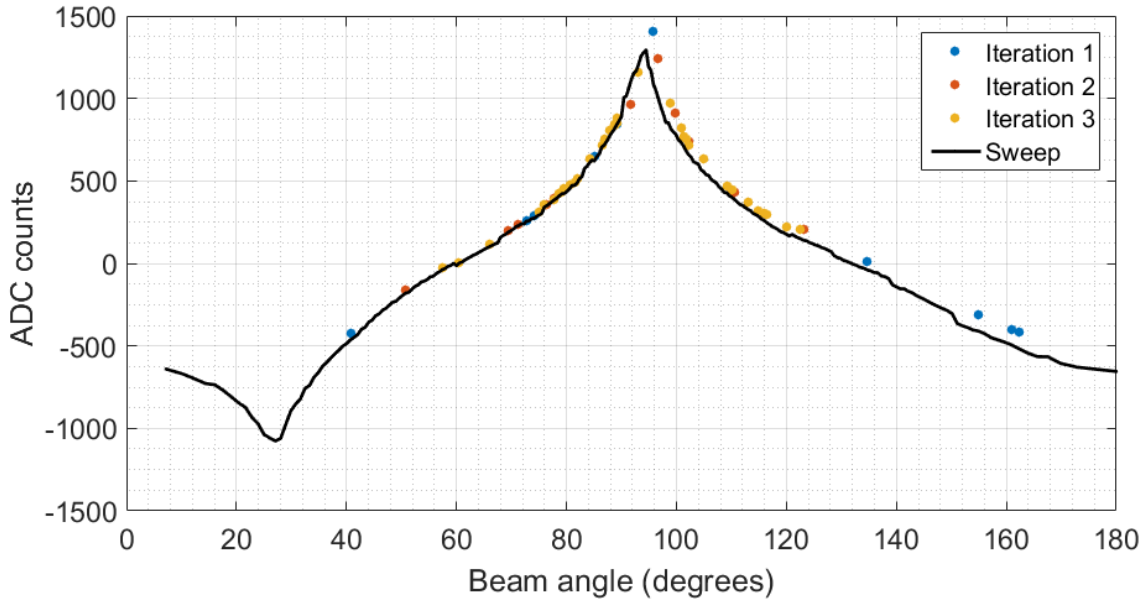


Figure 5.17: Result of an IWO test with the jammer broadside to the array, closer than previously. The tolerance remained as 2%. A total of 60 positions were tested. An exhaustive sweep of the same scenario is plotted as a line.

5.3.0.1 Two dimensional

One of the purposes of this research is to develop a system capable of localising an emitter from a UAV. This requires two dimensional searching to be efficient. An antenna array suitable for two dimensional searching was designed in Chapter 2. The chosen design is a square of four elements. It is capable of sweeping in two directions at 90° to each other. The linear array used in this chapter uses the same principle but only searches in one direction. Adding searching in a second dimension would be a relatively simple extension, but not necessary for this proof of concept.

There would be a few considerations required. For instance, the current algorithm searches until it finds the correct direction then stops. The second search could be performed after the first, but the time interval may make the solution less accurate. Alternatively the system could interleave searching in different directions, either at the iteration or the individual population member level. More interleaving would make the solution more accurate but would increase the amount of switching required in the hardware. More switching would increase the time taken (as the commands need to be sent and the components need time to settle). Experimentation would be required to find the optimum balance of speed and accuracy.

An alternative method would be to use the array designed in Chapter 2 but a different front end. A monopulse system is capable of forming one single null from multiple antennas (instead of two nulls in planes) and steering it. This method may be faster as both dimensions are measured at once. On the other hand it may take longer as there are more degrees of freedom, meaning the search space has more dimensions. This work is beyond the scope of this research, which is aiming only to create a proof of concept.

5.4 Conclusions and further work

In this Chapter two algorithms, Particle Swarm Optimisation and Invasive Weed Optimisation, were implemented. The aim of this was to increase the speed at which the system can determine the direction of arrival of an interference signal compared to an exhaustive sweep, by reducing the amount of time spent testing areas that were unlikely to contain the solution. It was found that after tuning, both algorithms had similar performance, with PSO sacrificing accuracy for speed compared to IWO. The IWO algorithm was then modified so that it could be used with existing hardware, and was tested in a real scenario. It was found that the algorithm was capable of determining the direction of the jammer after testing fewer positions than an exhaustive sweep: a sweep tests 256 positions, but the IWO algorithm could test as few as 60 positions depending on the scenario. Even when it was testing more positions than were required it still only tested 171.

5.4.1 Future work

In future the algorithm could be expanded. The current proof of concept has only searched in one dimension but a full system could use two dimensions, either searching in two orthogonal directions or by using a monopulse system.

Another extension could be the addition of attenuators. They may be necessary if the design of the array was changed. Attenuators would require significant rework of the algorithm as the optimisation is currently 'unconstrained'. This means that any combination of variables is allowed. However, if attenuators were added the optimisation would become 'constrained' [106]. This is because setting all the attenuators to very high attenuation settings would have the effect of decreasing the received signal strength, but would not provide useful information. To prevent this from happening the algorithm would have to restrict certain conditions, *constraining* the algorithm's operation. There are multiple approaches to tackling constrained optimisation. Some methods will increase the cost of certain states, thus effectively making them unattractive to the algorithm. This has the advantage of effectively returning the system to being unconstrained. This method does not always work and sometimes the problem becomes NP-hard, requiring significantly more computational time and effort.

Chapter 6

Methods for tracking moving emitters

In which: Realistic jamming scenarios are discussed . . . A method of implementing tracking is presented . . . Tracking bees and weeds are simulated and compared . . . Weeds are chosen as a clear winner

The Thesis up to this point has described the creation of a system that is capable of steering the beam of a phased array to find the strongest signal and therefore the direction of arrival of an interfering signal. The algorithm that carries out the locating was modified to increase the speed and efficiency of the search. This was successful, but the system still operates in the same manner as during an exhaustive sweep: it locates an interference source, then finishes the search.

In a realistic environment the apparent angle of arrival will, in all likelihood, change with time. This change may be only a few degrees due to small changes in the environment. It may be a steady movement in one direction if the source is mobile. If there is no line of sight link between the interference source and the receiver, it may be that the angle of arrival can change very quickly as reflections change. In this Chapter a method for allowing the system to efficiently track all types of changes is proposed.

In the proposed method, the algorithm reports its best position after each iteration, rather than running until a good solution is found. The algorithm uses prior knowledge of the previous best solutions to influence its search. It will react differently to small and large changes, ensuring solutions are found as efficiently as possible. The algorithm also reports the level of confidence that should be given to the solution, allowing a system to decide if it should be trusted.

In the previous Chapter, both PSO and IWO were shown to have similar performance when determining a single solution. As a result, both algorithms were modified to include tracking. The two algorithms were tested in simulation.

The simulations covered a number of scenarios. These included continuously changing solutions (ramps) as well as rapid changes (steps). These were tested using objective functions recorded in Sennybridge, a military training area in Wales, UK. The objective function was modified so that a second interference source could be added as a 'noise source' in any arbitrary position.

Following simulations it was found that the difference in performance between IWO and PSO is significant. The IWO algorithm was able to track both steps and ramps with a high degree of accuracy, achieving errors close to the theoretical minimum. It was also able to accurately determine when the solution could be trusted and when it was likely to be inaccurate. In contrast PSO was often inaccurate and, more concerningly, overconfident in the solution provided.

6.1 Literature review

It is important to first establish if tracking is actually necessary; it may be that in a typical scenario the jammer is unlikely to move. Considering the example of a jamming detection and mitigation system located at an airport, the threat of jamming might come from a nearby road. There are numerous cases of jammers being found in vehicles. In 2013 a vehicle carrying a low power, low cost jammer for privacy protection repeatedly disrupted the GPS system at Newark airport [14]. Other instances include a taxi driver in Australia using a low powered jammer to 'steal' fares [21], and on one road outside London up to ten incidents per day were recorded [117].

Indeed, vehicles carrying jammers has become so much of a problem that there are now multiple systems on the market that can detect them. Kar *et al.* designed a network of sensors to detect jammers on the road [40]. By combining static roadside detectors and mobile detectors mounted in cars, and correlating data at a central point, the vehicle carrying a jammer could be identified. An alternative approach requiring less infrastructure is the Chronos CTL3530 JammerCam [41]. This device captures an image of a passing vehicle carrying a jammer and relays the image to a server, allowing law enforcement agencies to identify the vehicle. The existence of multiple methods of detecting jammers on the road indicates that it is indeed a common enough occurrence that people wish to have methods to deal with it. The next consideration is if these jammers are capable of disrupting a sensitive site located some way from the road; after all, the incident at Newark Liberty airport may have been an anomaly.

Mitch *et al.* measured the signal characteristics of a number of commercially available jammers in [10]. A representative sample had their effective distance calculated. It was found that even a very low power jammer (43 mW in a 20 MHz band centred on L1) could prevent tracking in a receiver over 170 m away. At the other end of the scale was a jammer with a higher output (642 mW in the 20 MHz band), which could affect acquisition over 8 km away. Hence a vehicle carrying a jammer need not be close to critical infrastructure to impair its operation. Inspection of satellite imagery of major airports shows that the distance between the runway and a publicly accessible road is regularly less than 300 m, making it easily within reach of a low powered jammer. For example, the shortest distance between a runway and a road at Bristol airport is 300 m; Heathrow International is 211 m; at London Luton it is just 165 m. Given these figures, clearly it is likely that a small, low cost jammer will be able to affect the operation of an airport.

The second step in this work was to investigate existing methods of tracking moving emitters.

One method of tracking an emitter is to estimate the bearing, such as the method proposed by Lindgren [118]. They present a method in which both the position and velocity of a target can be calculated using multiple receivers on multiple bearings. This purely theoretical paper does not consider the practicalities of obtaining measurements. Bearing measurements, while useful, rely on consistent motion and a moving jammer may be on a curving road or may jump in position, depending on the environment.

Later on, Yiyu *et al.* proposed a method that removed the requirement for multiple receivers, and also allowed them to be static. By using the Time of Arrival (ToA) of a pulsed signal, the Direction of Arrival (DoA) and distance from the receiver could be calculated [119]. This method would have a high power consumption, requiring an accurate clock for timestamping of signals. However, it did demonstrate that it is possible to track an emitter from a single receiver.

Similarly, Rose patented a system for passive tracking of a moving emitter [120]. Both of these suffer from the same problems as previous localisation methods; namely, high power consumption and requiring certain signal properties (*e.g.* pulses) that are not associated with jammers. However, these papers do at least demonstrate that the tracking of moving emitters in real time is not only desirable, but possible. The next stage is to determine a low power method of tracking the interference source.

The optimisation algorithms presented in Chapter 5 provide a method of localising jamming that meets all of the criteria for this work. However, modifications will need to be made for the algorithms to be efficient when tracking.

Modifications to optimisation algorithms have been made before. The original PSO algorithm had an inertia term but no weighting function applied to it. In [121] a weighting term was added, and they also found that slight changes to the weight over time slightly improved performance. Other modifications include forming hybrid versions of algorithms to benefit from the advantages of each. For instance Bai *et al.* used a hybridised IWO and PSO in an array pattern synthesis problem [122]. This algorithm allows the 'bees' to reproduce in the same way as the 'weeds' of IWO, while still able to move around the search space. This overcomes the sensitivity of IWO to initial conditions, and the tendency of PSO to converge prematurely. If the algorithms as they are appear unsuited to tracking, it may be possible to improve the operation by combining them.

Thida *et al.* proposed a PSO-based 'tracking' algorithm for following an object through a crowded scene [123]. This method required a user to identify an object to track in the first frame of a video. The PSO would then find the most likely match to this object in subsequent frames, and coped well with changes in lighting and with occlusion. However, this was not true tracking, as it was not carried out in real time and the particles were re-initialised between frames [124]. This re-initialisation took the form of forcing the particles to spread away from their current position, taking into account the apparent motion of the object being tracked [125]. This method differs from the tracking algorithms proposed in this Thesis as a solution is not reported after every iteration; rather, it begins a new 'run' of the algorithm for each frame of the video and uses any previous knowledge to seed the start of its search. This method will only work if the emitter is moving at a relatively constant velocity, which may not be the case. In addition, the search is for a group of pixels against a background

that does not change significantly. When tracking jammers the environment is changing constantly, potentially making it a more difficult task despite the apparent simplicity of a one dimensional search.

The literature suggests that tracking is a worthwhile feature to add. While many have approached the task using computationally complex methods, others such as Zhang *et al.* have used simpler methods based upon modifying the starting positions of the particles using prior information obtained by previous runs. This idea is to be developed during this Chapter.

6.2 Adding tracking to optimisation algorithms

In a normal implementation of an optimisation algorithm, with each iteration the candidate solutions will converge on a good answer. They begin by *exploring* the search space and as the number of iterations increases, they move towards a small area in which it is likely to find a good answer. This second phase is called *exploitation*. Each different optimisation algorithm has a different method to control the movement of the algorithm between exploration and exploitation. For example, the particles in PSO begin spread out over the search space. As the algorithm progresses they will converge on a small area as the global best value leads them towards it. Similarly, the weeds in IWO will be spread out initially, but as stronger weeds reproduce more and weaker weeds die off, the tested positions will group together.

The proposed modifications will allow the algorithm to effectively rewind, moving from the exploitation stage (searching a small area for the best possible solution) back to exploration (searching a wide area for a likely minimum region). For the algorithm to recognise when it should be exploring and when it should be exploiting, there must be some way of recognising when the environment has changed. In this work the system will consider the best cost at the end of the iteration. If the cost has only changed by a small amount then the system should not need to search over a wide area; the environment has not significantly changed from previously. Hence the system should have some way of taking into account the magnitude of the change between iterations. It is anticipated that any algorithm will have four main modes of operation.

Initial optimisation When the program is first run, the distribution will be set so that the entire search space is covered. This means the system is not biased to any particular area. For the first few iterations the change in best cost will be large and the algorithm will continue to *explore*. However, as the best result is saved between iterations, as the algorithm progresses the best cost will become more steady. The change in best cost will shrink. The algorithm will move from the exploration to the exploitation phase. It will be necessary to ensure that the solution can be found even when the algorithm enters exploitation prematurely.

Static jammer When the target is a static jammer, the system will converge on a solution as with a standard optimisation algorithm implementation. The spread of particles will be small, allowing the system to operate in the *exploitation* phase and track peaks as they move with jitter or noise, but without unnecessary testing of positions a long way from the known good solution.

Moving jammer As the jammer moves, the peak in the cost function will move. This will cause the cost value for the current best weed to decrease as it is no longer in the centre of the peak. This change in maximum will cause the spread of particles to increase. A slowly moving jammer will only lead to a small increase in the range, meaning the system only searches near the existing peak and remains in *exploitation* mode.

Step change A single jammer is unlikely to move suddenly. However, if there is no line of sight path between the jammer and the receiver, the strongest multipath route could change (especially if the jammer was moving). The large change in best cost will cause a large increase in the spread of particles, returning the mode of operation to *exploration*.

The modifications will use principles of closed-loop control systems from control theory [126]. The difference in best costs between generations will be considered the set point and the system will always try to maintain this at zero. If the best cost changes between generations this is considered a disturbance which the control system will attempt to remove. It will do so by reverting back to an exploration state so that a new, good solution can be found.

For the tests in this Chapter, a range of cost functions were generated. They were based on data recorded at Sennybridge, so considered to be representative of a real environment. The data was then stretched using interpolation so that there were 360 points (one for each degree of a circle, to make it easier to visualise). The data were then cyclically shifted, one degree at a time, with each step being saved in a different file. This meant the Python test program could generate a numerical position (either randomly generated, calculated mathematically, or input manually into a test vector) for the solution, and open a file where the solution is at that position. Additional functions were written to allow the programs to open new cost functions at runtime, allowing the objective function to be changed dynamically. However, the large number of file operations required makes these simulations slow and so all discussions of speed in this Chapter will be in terms of iterations or of positions tested, and not absolute time taken.

The code for both IWO and PSO optimisation algorithms has been modified so that the algorithm will not loop indefinitely. Instead it will carry out just one iteration and then 'pause' while new cost functions are opened. This new program flow means that the objective function can be opened as often as once per iteration. The section that updates the objective function is also able to open a second file. The two objective functions can be summed with their relative magnitudes controlled. This means the system can simulate a reflected (noise) signal with a lower magnitude than the 'true' answer, allowing more extensive testing of the the algorithm. It is assumed that the system is linear and if two jammers are present the resultant objective function is a linear combination of the two scaled cost functions.

6.2.1 Tracking Invasive Weed Optimisation algorithm

In a standard implementation of the IWO algorithm the key feature that allows it to successfully and efficiently locate a solution is the change of the 'range' with the iteration number (as shown in (5.3) in Chapter 5). The range is wide initially, narrowing as the algorithm progresses and is assumed to be closing in on a solution. To adapt IWO to tracking the range

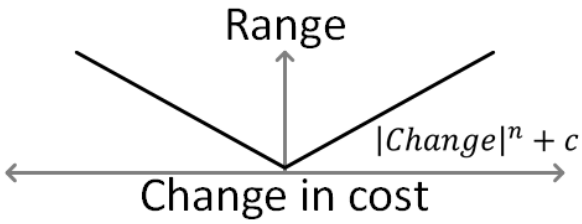


Figure 6.1: Graph showing how the range of the weeds spreading would change with changing error.

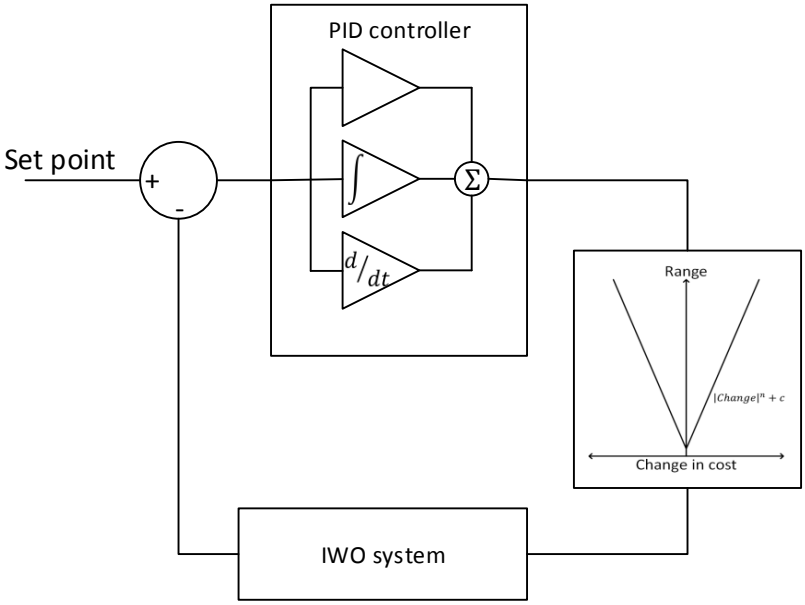


Figure 6.2: Block diagram to show how the PID controller and range adjuster fit into the IWO system.

equation was modified. Instead of the range being set by the iteration number, it was linked to the change in best cost between generations.

In IWO, 'range' refers to a single number, which determines how far from a parent plant a seed is allowed to be placed. The original IWO algorithm used this range as the standard deviation of a normal distribution. Tests were run to compare a normal and a uniform distribution and it was found that a normal distribution did indeed perform better. Hence the seed will be placed at some distance from the parent weed, with the exact position assigned probabalistically according to a normal distribution.

The modification proposed to allow tracking is as follows: it will be based on a standard Proportional, Integral, Differential (PID) controller. In this system the input (which it is trying to maintain at zero) is the change in best cost. The output (the thing which can be changed by the controller) is the range. If the solution is not moving, the best cost will not change significantly and the range should remain small. If there is a large change in best cost then the space should be searched again, so the range should be made large. Fig. 6.1 shows how the range and cost would be connected; the cost can change both positively or negatively to cause a change in range. The values of n and c, and therefore the shape of the transfer function, are to be determined later. Initially only proportional control will be implemented. If this is found to be insufficient then more functions can be added in future.

A complete system is shown in Fig. 6.2. The controller measures the change in best cost and, using the transfer function from Fig. 6.1, creates a new range for the IWO algorithm. The IWO algorithm carries out one iteration of searching, and reports back a new best cost. This operation means that the system is not exactly like a control system, because there is no predictable link between the range and the change in cost. However, the overall principles are the same. The only exception is that in this new system, the range will only be a function of the change in best cost, and will no longer be connected to the iteration number.

6.2.2 Tracking Particle Swarm Optimisation algorithm

Both IWO and PSO had similar performance during the static tests presented in Chapter 5. Hence the PSO algorithm will also be modified as there is no reason to choose one over the other based on their performance.

As before, a method for changing the algorithm's mode of operation, from exploitation back to exploration, needed to be designed. In PSO there is no explicit link between the iteration number and the spread of particles (and therefore the exploration/exploitation mode). Instead the speed of the particles has a tendency to decrease as the number of iterations increases, leading the system to begin exploiting more promising areas. Hence the proposed method is to apply a random boost to the velocity if a change is detected. This boost increases the speed of particles, making the algorithm behave as it did earlier in the algorithm's operation, when exploration was encouraged. Once again the disturbance applied will be proportional to the change in the cost.

Two modifications were made to the basic PSO algorithm to accommodate the moving solution. The first change modifies the velocity equation. If the answer is deemed to have changed, a random extra velocity will be applied to each particle. The change in best cost is used to generate a maximum velocity. The equation for generating this velocity is the same as that used in the range calculations for IWO, so that a change in cost of 50% will result in the maximum velocity being increased to 180°. A random number is then generated between ±(Largest magnitude). This velocity is applied to each particle to encourage it to explore a wider area again. As the sign can be positive or negative the particle may move in either direction, allowing all of the search space to be covered if necessary.

The second change is in relation to the two values used to help determine the velocity: the 'Local Best' and the 'Global Best'. As explained in Chapter 5, 'Local Best' is the best value seen by a given particle up to this point. 'Global Best' is the best value seen by any particle in the swarm. If the cost function changes (that is, the jammer moves) then those bests that are stored are no longer guaranteed to be the best. Hence if a change in solution is detected, 'Local Best' and 'Global Best' are reset.

In the static case presented in Chapter 5, the global best `SwarmBestCost` and iteration best `Bees[0].Cost` are the same. However, in the case of a moving solution, they may differ. It may be the case that a better solution was found in a previous iteration, and hence `SwarmBestCost` will not be updated. This is only the case if the change in best cost between iterations (as measured using `Bees[0].Cost`) is less than 5%. Changes greater than 5% will trigger `SwarmBestCost` to be reset. In tests, results using both `SwarmBestCost` and `Bees[0].Cost` will be discussed.

It was found that if the algorithm was set so that reduction in best cost caused a reaction, the system became overly sensitive and it would never attempt to exploit an area; it would get stuck in the exploration mode. As a result it was programmed so that a 5% *change* (instead of a reduction) in best cost would trigger the change in behaviour.

6.3 Results

A number of different tests were run to assess the algorithms' ability to track. The first test was jumps in position. The second test was a ramp. The third test considered what happened with steps and ramps but with an additional 'noise' signal that would either track the true signal, or move around, varying both in position and amplitude.

6.3.1 Step tests

For the step tests, the position of the solution was initialised to 45°, which was considered the 'home' position from then on. The position of the solution would change by increasing amounts (starting with a step of 10° and increasing by 20° each time) before returning to the home position. Each position was held for ten iterations to allow the system to settle.

6.3.1.1 Step tests: Invasive Weed Optimisation

The graph in Fig. 6.1 can be expressed as an equation (6.1). The values of a and c were held constant for all tests in this section. They were chosen so that if the change in error was zero, the range was four. This allowed for updating of the position due to jitter without testing positions unnecessarily. They were also set so that if the best cost is halved the range is set to 180, meaning the entire space can be searched when there is a significant change.

$$\text{Range} = a|\text{Change}|^n + c \tag{6.1}$$

One run of this test is shown in Fig. 6.3. In this example a normal distribution was used and $n = 1$. The 'correct' answer is the point with the best cost read from the objective function, and is plotted in purple. Every time it changes, a new objective function is opened. After each iteration the algorithm would report the best position it had found. This is plotted in yellow. All other positions tested were also saved, and they are shown in grey. The envelope of all tested positions is shown for interest only. Note that the envelope can be misleading when the points wrap, as the spread appears larger than it is.

When the correct answer is not changing (the flat portions of the purple line) the range is very small. It can be seen that all the positions tested are close to the answer. However, when the answer changes the positions tested spreads out. This indicates that the range value has increased. When the step change is small (as in the steps at 10 and 20 iterations, which are steps of 10°), the range increase is small, but when the step is larger the range increase is also correspondingly larger. This shows that the control system is working as intended. The algorithm has a lag as it takes one iteration to 'notice' that the position has changed. However, it generally finds the solution after one further iteration.

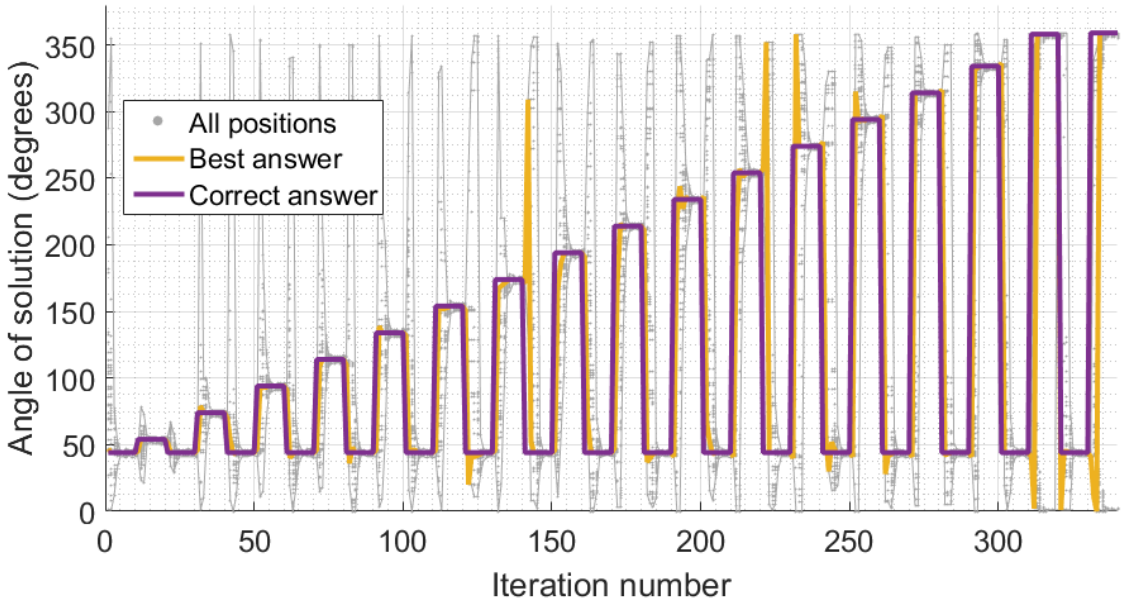


Figure 6.3: Graph showing the modified IWO system successfully tracking step changes in position. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

Table 6.1: Table showing the effect of changing the transfer function shape on speed and error of the IWO algorithm when tracking step changes. Each run was 340 iterations. Percentages are quoted to two significant figures.

Distribution type	n	Positions tested	Reported correct	Solution correct	$\leq 2°$ error
Uniform	1	29631	241 (71%)	242 (71%)	283 (83%)
Uniform	2	28465	237 (70%)	227 (67%)	264 (78%)
Uniform	4	28236	242 (71%)	232 (68%)	281 (83%)
Uniform	6	28175	240 (71%)	233 (69%)	283 (83%)
Uniform	0.5	29217	237 (70%)	238 (70%)	274 (81%)
Uniform	0.25	30567	243 (71%)	257 (76%)	301 (89%)
Uniform	0.167	30891	243 (71%)	247 (73%)	287 (84%)
Normal	1	29381	227 (67%)	229 (67%)	267 (79%)
Normal	2	28816	228 (67%)	223 (66%)	269 (79%)
Normal	4	28668	226 (66%)	225 (66%)	266 (78%)
Normal	6	27490	223 (66%)	216 (64%)	253 (74%)
Normal	0.5	28887	224 (66%)	218 (64%)	253 (74%)
Normal	0.25	28307	216 (64%)	214 (63%)	251 (74%)
Normal	0.167	27315	219 (64%)	198 (58%)	236 (69%)

The yellow line appears to show the system struggling when the step change is very large. In reality this is an artefact of the system wrapping - as 360° is the same as 0°, a solution of 5° is close to the actual answer of 360°, and they are simply displayed differently.

For the test shown in Fig. 6.3 the value of n in was 1. Tests were run to discover the effect of changing the value of n on the system performance. Varying n changes the rate at which the algorithm reacts to small and large changes. This would manifest as the envelope enclosing all positions growing or shrinking when a change is noticed. In terms of performance, the effect is seen as a variation in how many positions are tested, and the average error over time.

It is also possible to vary the type of distribution used when generating the random numbers. Table 6.1 shows the effect of changing n and the distribution type. The total number of positions (which affects the speed at which the algorithm runs) and the average error are listed, as well as the number of iterations where the best answer according to the algorithm was correct, or within two degrees of the correct answer. The results contained in the Table are after one run of each setting.

Table 6.1 implies that a uniform distribution performs better than a normal distribution based on the lower mean error. This is in contrast to the static tests, where the opposite was true. The average number of positions tested was 3% higher for the uniform distribution; however, the average error was 12.8% higher for the normal distribution. Hence by using a uniform distribution there is a large improvement in the solution for only a small increase in time taken. The average number of positions tested per iteration ranges from 80 to 90. This is the same as with static tests, where the average number of positions tested was approximately 86 for a normal distribution and 93 for a uniform distribution. It is significantly lower than an exhaustive sweep, which would require 256 tests per iteration.

Depending on the exact implementation, the solution could be absolutely correct up to 76% of the time (uniform distribution, $n = 0.25$). Given that each position is held for ten iterations before it changed, having a correct answer for 90% of the time (or 306 of 340 iterations) means it is able to settle to the new value after a step change within one iteration. A correct answer 80% of the time means it is able to settle within two iterations of the change. The proportion of the time spent within 2° of the solution is higher than the time spent with the exact right answer, as expected. This error (within 2°) was chosen as it is close enough to the correct solution for effective localisation or mitigation.

The average error can appear high as when a step change happens there is a lag of one iteration before the change is noticed. At this point the error will be very large. If an error was 180° (the largest possible) but it settled within one iteration, the average error would still be 18 (error of 180 divided by ten iterations). Hence this measure is not as useful as the number of iterations in which the answer is correct in this test type. To summarise the results, the mean error for a uniform distribution was between 13.8 and 14.4, while for a normal distribution it was between 14.9 and 17.5. Overall the error for a uniform distribution is smaller, most likely due to it converging on new positions after a step more quickly than the normal distribution.

The algorithm will report that a solution has been found when the cost is within 5% of the previous best cost. The 'Reported correct' column gives a count of how many iterations the algorithm reported that it had found a correct answer. When compared to the 'Solution

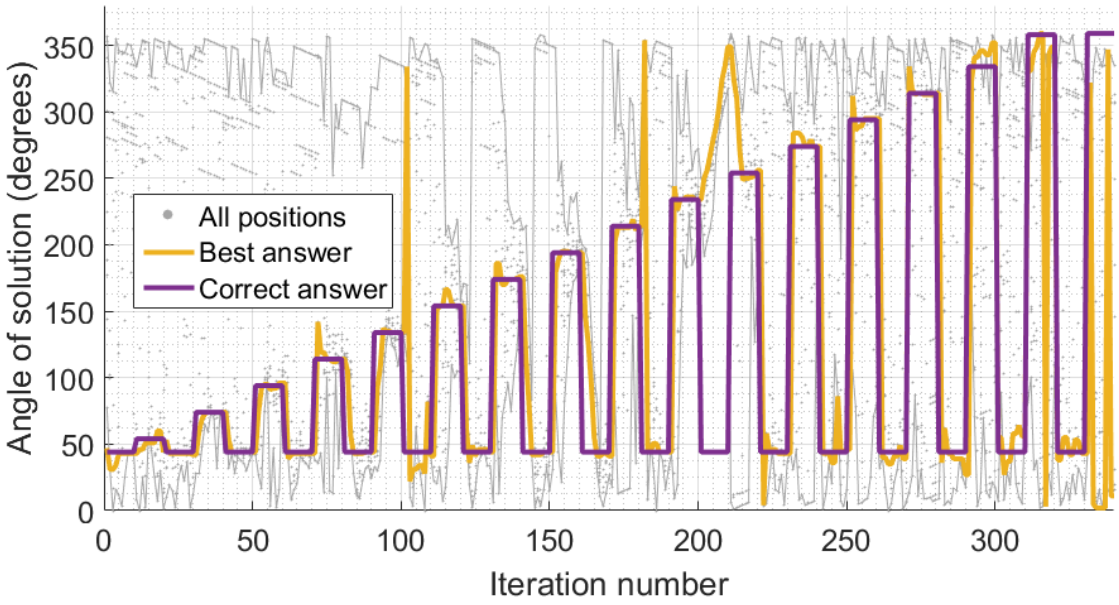


Figure 6.4: Result showing modified tracking PSO algorithm when the correct answer includes step changes of increasing size. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

correct' column, an indication is given of the under- or over-confidence of an algorithm. Both uniform and normal distributions generally report that they are correct slightly more often than when there is no error. Tuning of the threshold (currently at 5%) will affect the confidence of the algorithm, but the values reported in Table 6.1 would imply that the correct value is close to 5. Further work may involve the modification of this reporting to be more fuzzy, so that it can give a range of confidence values instead of just zero or one.

6.3.1.2 Step tests: Particle Swarm Optimisation

The step test was carried out using both uniform and normal distributions when generating random velocities. They were also run saving different parameters as the 'best cost' for each generation.

Fig. 6.4 shows the result of a test with a uniform distribution. After each iteration the value of `SwarmBestPosition` was saved. In this test, over 340 iterations it found the correct answer only 52 times, or approximately 15% of the time. This is in contrast to the tracking IWO algorithm which found the exact correct answer 242 times, or 71%, in an equivalent test. It is also clear from Fig. 6.4 that the system is not moving between exploring and exploiting appropriately. When the step changes are small the range of positions tested is still very large (note the points scattered over a wide range between iterations zero and 100). Later it does appear to change behaviour but significant overshooting is observed.

Table 6.2 gives the results of a number of experiments testing the PSO's ability to track steps. The population was initially set to ten, but all tests were then repeated with a population of 20 when it was clear that the algorithm was not performing well. The table gives the average error for each test, and the number of positions for which the answer was correct or within 2° of the answer.

Table 6.2: Table showing the effect of changing the distribution type, the best answer saved and the population of a PSO algorithm when tracking step changes. Each run was 340 iterations. Percentages are quoted to two significant figures.

Distribution type	Variable saved	Population	Average error	Reported correct	Solution correct	$\leq 2^\circ$ error
Uniform	SwarmBestCost	10	23.7	211 (62%)	58 (17%)	155 (46%)
Normal	SwarmBestCost	10	56.3	201 (59%)	33 (9.8%)	78 (23%)
Uniform	Bees[0].Cost	10	28.5	184 (54%)	23 (6.7%)	75 (22%)
Normal	Bees[0].Cost	10	10.2	212 (62%)	39 (11%)	113 (33%)
Uniform	SwarmBestCost	20	30.4	234 (69%)	113 (33%)	190 (56%)
Normal	SwarmBestCost	20	24.4	252 (74%)	106 (31%)	202 (59%)
Uniform	Bees[0].Cost	20	11.2	252 (74%)	60 (18%)	176 (52%)
Normal	Bees[0].Cost	20	9.98	241 (71%)	65 (19%)	154 (45%)

The average error is quite variable; results were obtained with average errors from as low as 9.98 (a very good result), to as high as 56.3. It was noted previously that the average error can appear suprisingly high during step tests due to sudden large changes skewing the results. However, this explanation does not account for the fact one test had an average error of 56.3° - far larger than expected. It also does not explain why one test achieved an average error of just 9.98°, which is better than should reasonably be expected. The very high error is due to the algorithm not detecting that a step has occurred, such as between iterations 200 and 215 in Fig. 6.4. This is a common occurence in the PSO algorithm, despite the method of detecting changes being the same as for the IWO algorithm. The very small error is possibly related to the same root. The PSO algorithm often identifies changes when there are none, meaning the particles do not converge. If they remain spread out at all times, when a large step change in the solution occurs, there is likely already a particle nearby. Thus, by happy accident, there is a new best solution that is close to the correct answer and overall the average error is reduced.

While the average error values give a picture of an algorithm that is (sometimes) able to cope well with rapidly changing positions, the final two columns of Table 6.2 paint a different picture. These columns show the number of iterations in which the algorithm produced an answer that was exactly correct, or within 2° of the answer. The best result only found the correct answer 33% of the time, and most were correct less than 20% of the time. The amount of time spent within 2° of the error is low, with only three tests finding a good solution more than 50% of the time.

It is also interesting to note at this point that the average error of a PSO algorithm, and its precision in finding an answer, had little correlation with each other. The most successful PSO test in terms of average error (9.98%) was only within 2° of the answer 45% of the time, while a test that was close to the correct answer 59% of the time had a much higher average error of 24.4. This may indicate that when the average error is worse it is managing to transition between exploration and exploitation, leading to a higher average error due to the sudden changes, but the algorithm is still managing to find the correct answer more often.

Increasing the population from ten to 20 did have the expected effect in general. The average error decreased and proportion of correct solutions increased for every combination of variables. It also increased the number of positions tested (6800 instead of 3400). This implies that the poor results were as a result of the low population. It may have been possible to increase the population further and obtain results close in quality to those from the IWO algorithm. However, the fact remains that the PSO algorithm is poor at moving between exploration and exploitation and perhaps is not suitable for tracking.

One last factor that counts against the PSO algorithm is its overconfidence. One test had it find an answer within 2° of the solution just 78 times (23%), in the same run it estimated that it had found a good solution 201 times. This was the case in every run: that the PSO algorithm would dramatically overestimate its success in finding a solution. All but one tests reported that the solution was correct more than 55% of the time. It is important that an algorithm is able to decide if it has confidence in a solution or not, to allow other systems using the result to decide whether to trust it if there is conflicting information.

6.3.1.3 Step tests: comparison

It was noted in Chapter 5 that generally, PSO tested fewer positions and achieved a slightly worse solution than IWO, which tested more positions but achieved a better answer (one that is closer to the correct answer). This trend is the same in these new step tests, but with a much bigger difference between the algorithms.

In step tests, the IWO algorithm tested as many as 90 positions per iteration, but would be correct more than two thirds of the time (apart from in a few tests). The PSO algorithm would test only 10 or 20 positions per iteration (depending on the population setting). However, the solution was, more often than not, incorrect.

The best IWO test found the correct answer 76% of the time (257 iterations), and was within 2° of the solution 301 times, or 89%. The best PSO test found the correct answer 113 times, or 33% of iterations. The same test found a solution within 2° of the correct answer 56% of the time. The theoretical best achievable is a good answer 90% of the time, assuming that the algorithm reacts to a change after one iteration. Hence it is possible for IWO to operate near the theoretical best, while the PSO is performing poorly in comparison.

The IWO algorithm does not give a correct answer for iterations when there is a step change, as expected. However, it will converge quickly on the new solution, typically within one or two iterations. In contrast, the PSO algorithm is sometimes better at reacting to large changes because the particles have not converged, but it also has significant errors when the position is not changing because it does not converge on them.

The results from these step tests imply that the PSO algorithm is not able to produce a good answer in one iteration. This is discussed further in Section 6.3.3.3, where a more detailed discussion of the implications is held. However, the overall result is the PSO algorithm is one that is capable of very quickly reporting a result that is unlikely to be correct, while IWO takes longer but produces answers that are more reliable.

Both algorithms also report their confidence in the solution after each iteration. This is because the algorithm will report a solution after each iteration, but a system using this information may need to know if it is likely to be correct. For example, if the angle of arrival of the source was used in addition to other sensors, there may be disagreements in the result so if the solution found by the algorithm is not trustworthy, it can be discarded.

To understand how well the algorithms are recognising their own errors, one test for each algorithm was examined, chosen at random. The IWO test used had a normal distribution and $n = 1$. It reported that it had found an answer 227 times. It in fact found the correct answer 229 times, and was within 2° 267 times. There are 47 iterations (14%) where the confidence of the algorithm was misplaced, whether that was a false negative (reporting that the solution is unreliable when it is correct) or a false positive (reporting that a solution is good when it is not correct). The PSO test used had a uniform distribution, a population of 10 and saved SwarmBestCost. It found the exact answer 58 times, and was within 2° 155 times. The algorithm assigns the wrong confidence 56% of the time, which is 191 iterations. Hence the PSO algorithm is not only significantly less accurate than IWO, but it is also unreliable in its assessment of the quality of the answer.

6.3.2 Ramp tests

It is assumed that, at a location sensitive to vehicle-based GPS jammers such as an airport, the most common scenario is a jammer moving in one direction. It is assumed that if the system is able to track a given ramped signal, say 5° per iteration, it will also be able to track a slower moving target.

The test vector in this Section begins with the correct answer at 0° (or 360°). With no settling time allowed, the algorithm ramps at the rate of 10° per iteration. This was chosen as a ramp speed that was unlikely to be exceeded.

6.3.2.1 Ramp tests: Invasive Weed Optimisation

The first version of this program left the code unchanged from the previous step tests and simply considered the change in best cost between iterations. However, it was found that the system could struggle to keep up with some ramps. To understand the problem, imagine that the solution has remained in the same place for some time. The change in best cost is zero and so the range is at a minimum (four). If the answer moves by less than four degrees the best answer will be found and so the best cost will not change. However, the new best position is not in the centre of the parent weeds. If the next change is greater than four the solution will be missed, but only by a small amount. The following change may not be within reach of the seeds, and so the solution drifts further away from the reported best position. To mitigate this problem, an additional test was added to the code: if the change in best cost is very small, but there is a significant change in the best position (three degrees or more) the range is increased slightly. This significantly improved the system's ability to track ramps and removed the oscillation-type shape to the range value with steeper ramps.

Fig. 6.5 shows the result of a ramp test. The angle of the solution changed by ten degrees per iteration, and it moved from 0 to 360°. As with the step tests in Section 6.3.1.1 above, the correct answer is shown in purple, the best answer according to the algorithm in yellow, and all positions tested are in grey. The graph shows that the algorithm initialises with weeds spread over the whole search space, but it quickly narrows the search to the vicinity of the correct answer. It is then able to track the correct answer consistently for the rest of the test. The weeds test a range of positions near to the correct answer and so are able to keep up with variations, but do not explore unnecessarily.

Table 6.3 gives all the results for a ramp of 10 degrees per iteration. n refers again to the exponent n in (6.1). Once again the number of iterations for which the error is zero and when the error is less than 2° are given. The average error is quoted as this is a useful metric in these tests, unlike for the step tests in Section 6.3.1.1. Note that there are far fewer iterations (36, instead of 340) in these tests than the step tests, resulting in a much lower number of weeds tested.

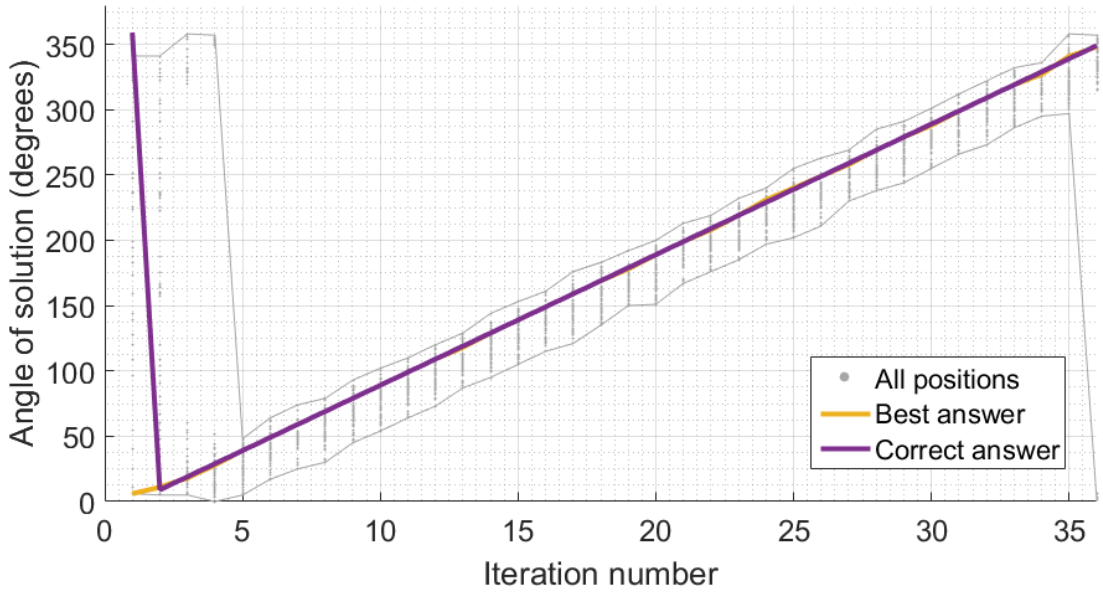


Figure 6.5: IWO algorithm tracking a constant ramp. A uniform distribution was used. $y = x$, or $n = 1$ (*cf.* Fig. 6.1). The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

Table 6.3: Table showing the effect of changing the transfer function shape on speed and error of the IWO algorithm when tracking a ramp. There were 36 iterations.

Distribution type	n	Positions tested	Average error	Reported correct	Solution correct	$\leq 2^\circ$ error
Uniform	1	2296	0.67	34 (94%)	22 (61%)	35 (97%)
Uniform	2	2313	0.94	34 (94%)	17 (47%)	34 (94%)
Uniform	4	2219	0.83	34 (94%)	16 (44%)	35 (97%)
Uniform	6	2239	1.83	33 (92%)	17 (47%)	31 (86%)
Uniform	0.5	2300	0.56	34 (94%)	22 (61%)	35 (97%)
Uniform	0.25	1796	38.6	15 (42%)	3 (8.3%)	6 (17%)
Uniform	0.167	1896	48.2	7 (19%)	1 (2.8%)	3 (8.3%)
Normal	1	1944	2.06	32 (89%)	8 (22%)	25 (69%)
Normal	2	2321	1.00	32 (89%)	23 (64%)	33 (92%)
Normal	4	2383	0.64	34 (94%)	19 (53%)	35 (97%)
Normal	6	2350	0.69	34 (94%)	19 (53%)	35 (97%)
Normal	0.5	1801	21.4	15 (42%)	2 (5.6%)	8 (22%)
Normal	0.25	1852	55.14	4 (11%)	2 (5.6%)	2 (5.6%)
Normal	0.167	1829	40.39	6 (17%)	3 (8.3%)	6 (17%)

The average number of weeds tested per iteration varied from 49 to 66. This is significantly lower than during steps, when between 80 and 90 positions were tested per iteration on average. This may be due to the way in which the number of seeds is assigned. When a step test was run, a significant portion of the time was spent with the answer unchanging. During this time the range was small and so many of the weeds had a very good solution. Hence when assigning seeds, many of the parent weeds will have produced a large number of seeds. On the other hand during a ramp test, the correct answer is further away from many of the parent weeds (due to the solution changing from when the parent was first created and when it was tested) and so many weeds are assigned a lower number of seeds. At this time the algorithm is not working as well as it could as it is not taking into account overcrowding; a solution to this will be discussed later.

Table 6.3 shows that several combinations of distributions and values of n will create a system that will produce an answer close to the correct answer almost every time. The algorithms typically take two or three iterations to converge and so do not achieve 100% accuracy as they miss the very beginning of the ramp.

The algorithm reports its confidence in the solution as before; if the best cost has changed by less than 5% between iterations, the solution is marked as reliable. When this algorithm is initialised, the previous 'best costs' are created as arbitrarily large. Hence for the first two iterations the change in average best cost will be very large until this initialisation is shifted out of the algorithm's memory. As a result the algorithm will mark the first two answers as unreliable, meaning it cannot report more than 34 of 36 answers as correct.

When the algorithm is performing well and successfully tracking the solution, it is accurate in determining how often the solution is correct. However, when the algorithm performs poorly (such as when the value of n is less than one), it becomes overconfident, reporting 7 correct answers for a uniform distribution when $n = 0.167$, when in fact it was only within 2° three times. The gradual change in environment causes only small, if constant, changes in the best cost. Some of those changes may be under the 5% threshold and cause the algorithm to be erroneously confident.

It is most interesting to note the effect of having $n < 1$. The table shows very large average errors of up to 55 ($n = 0.25$, normal distribution). The reason for these errors is shown in Fig. 6.6. The insensitivity of the algorithm to small changes leads to a drift away from the correct answer with time. When the difference is sufficiently large the best cost will change suddenly, allowing the range to expand and in some cases, allowing the algorithm to re-find the correct answer. When the solution is wrong, the algorithm is able to determine that it should not be trusted. A less steep ramp of 4° per iteration was successfully tracked by all values of n.

This sort of systematic error, or change in best cost, could be overcome by implementing the integral part of a PID controller. The systematic error would build over time and increase the change in cost seen by the range calculator, allowing gradual errors to be corrected.

Generally, higher powers produce a smaller average error but test a wider range of solutions. The wider range increases the number of positions tested and therefore the time taken. The higher power increases the range when even a small change occurs. For some powers the system could be seen to be 'overreacting'. The effect can be seen in Fig. 6.7. All three graphs

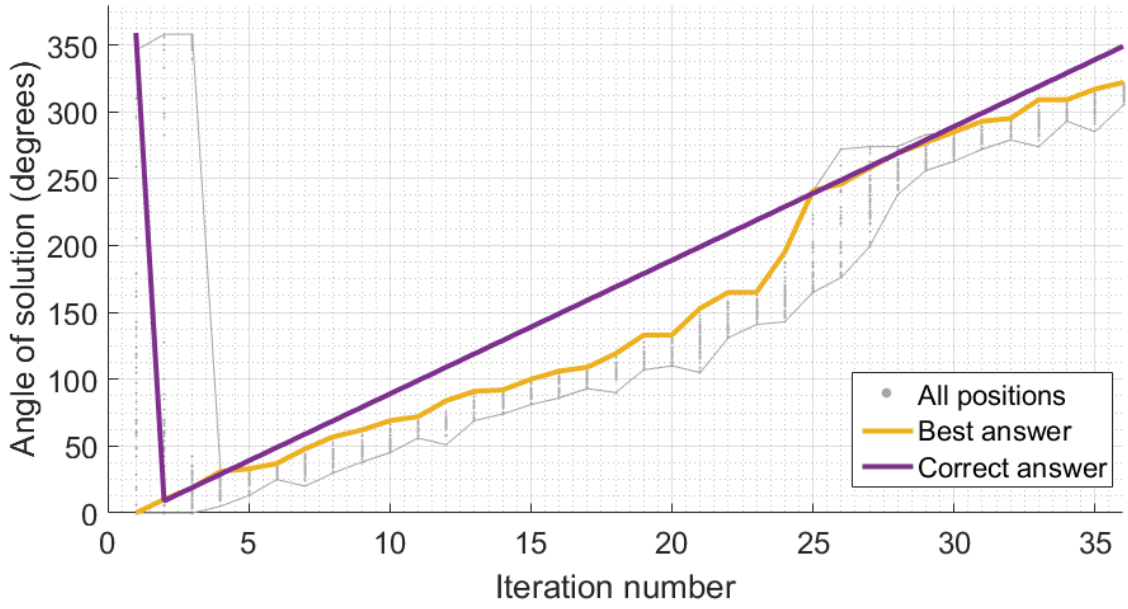
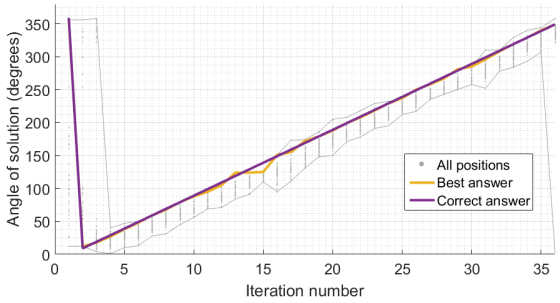
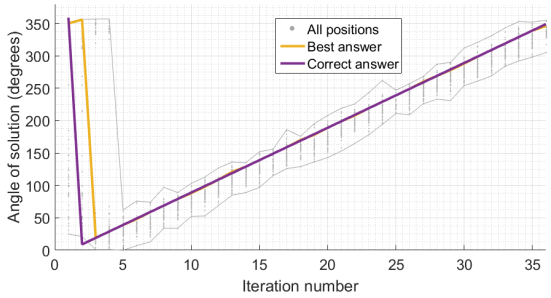


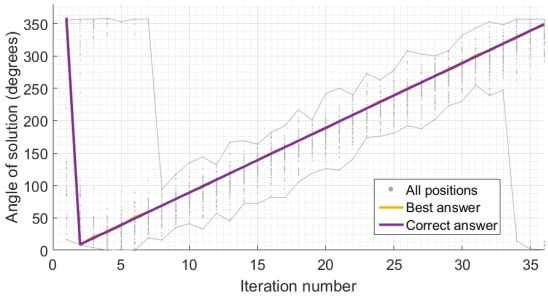
Figure 6.6: Tracking a ramp using the IWO algorithm with a uniform distribution and $n = 0.167$ (*cf.* Fig. 6.1). Note the sudden change in range around iteration 23. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).



(a) $y = x$



(b) $y = x^2$



(c) $y = x^4$

Figure 6.7: Comparison of different values of n (*cf.* Fig. 6.1) for IWO. All produced using a normal distribution, tracking a ramp of 10° per iteration. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

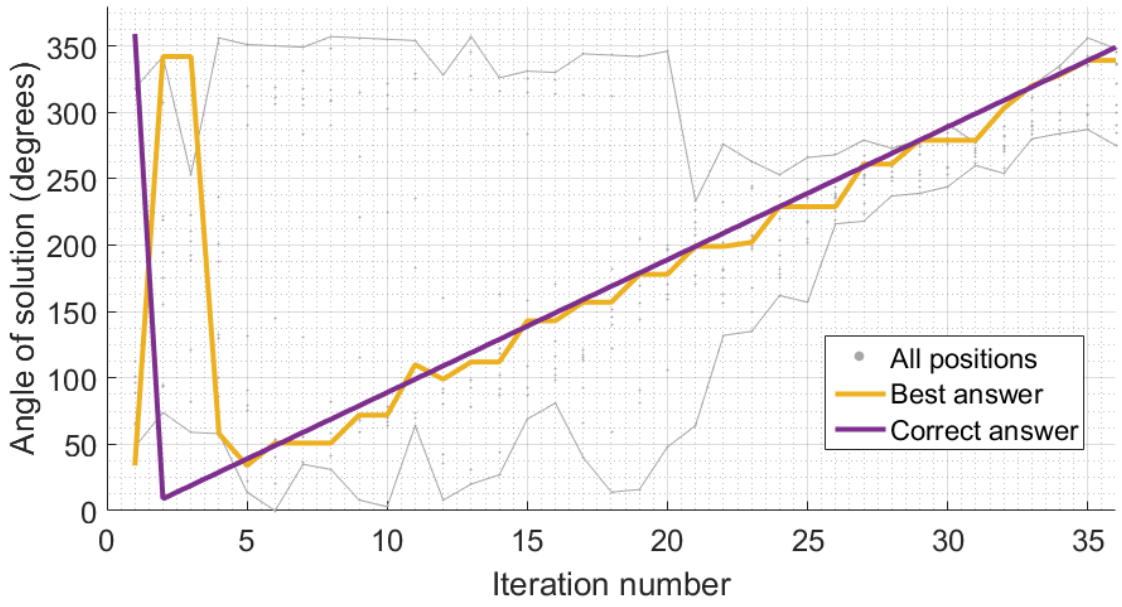


Figure 6.8: Result of using the PSO algorithm to track a ramp. It used a uniform distribution and recorded `SwarmBestCost`. The slope was 10° per iteration. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

show results from tests using a normal distribution, with varying values of n. The slope of the ramp is 10° per iteration. Figs. 6.7a, 6.7b and 6.7c shows when $n = 1$, 2 and 4 respectively. In all three cases the algorithm (yellow line) is able to track the changing correct answer. When $n = 2$ the range of positions tested is slightly greater but the answer is also correct more often (within 2° 92% of the time, compared to 69% of the time for $n = 1$). Increasing n to 4 results in a slight increase in correct answers (97%) but the number of positions tested is also increased (2383, compared to 1944 for $n = 1$ and 2321 for $n = 2$). The spread of grey dots (all positions tested) shows that increasing the value of n increases the range when a small change is detected, which leads to more positions being tested.

6.3.2.2 Ramp tests: Particle Swarm Optimisation

The same test vector was used to test PSO with ramps as was used for IWO, to allow comparison between algorithms. For some tests, however, the relatively poorer performance of the PSO algorithm meant that a less steep ramp of 4° per iteration was used so that at least some data could be obtained.

Fig. 6.8 shows the PSO algorithm tracking a ramp with a slope of 10° per iteration. The 'best answer' given is the value of `SwarmBestCost` after each iteration. As a result there are occasions (such as iteration three) when the best solution is not a position that was tested in that iteration; the best cost was found in a previous iteration and it was not improved upon. When presented with a ramp the PSO algorithm is able to track it with some effectiveness.

The range of positions tested is generally small. The envelope is somewhat misleading as there are just one or two particles that remain far from the rest of the group. Later on these particles also converge. The main group of particles does track the correct answer but the precision is low. The best answer found is rarely on the line. Instead it appears to step: the correct solution will be found and this will remain the best answer for several iterations

(despite the correct answer moving away) until a new best is found. The imprecision is caused by the particles being widely spread. Often the particles are a long way from the correct answer as they have not converged as well as they could. As a result the number of times the algorithm finds a good answer is very low.

The phenomenon of particles a long way from the correct solution having a low velocity is observable in both this ramp test and in previous step tests, such as in Fig. 6.4. It is assumed this is to do with the clipping of the particles' velocities. A particle that is a long way from the global best will experience a strong pull towards it, and therefore have a high velocity. This velocity is then clipped to the maximum, which is 256. As a result the particle is only able to move by one position each iteration. This assumes that the random number generated for the global velocity term is large. When a small number is generated the velocity is lower, which explains the occasional times when the low velocity particles move suddenly.

Table 6.4 shows the results of a number of different ramp tests. Slopes of both 4° and 10° were tested. As the test was run until a full 360° had been covered, the number of iterations varied. If the slope was 4° there were 90 iterations; if the slope was 10° there were 36. The population was kept as 10 for all of these tests. This means the total number of positions tested was either 360 or 900, for slopes of 10° and 4° respectively. Additionally, percentages are given in the final two columns to allow comparison, as well as the number of iterations for which the algorithm reported that it had found a good answer.

Table 6.4 shows a similar story to the step tests. The PSO algorithm is not particularly adept at tracking. The average error is quite high: between 8.10 and 13.1. Unlike the step tests there are no large jumps during which the algorithm can reasonably be expected to be wrong, and a good error should be small. Saving `SwarmBestCost` generally results in a worse answer (in terms of average error) than saving `Bees[0].Phase1`, but not always.

The algorithm produces a high average error, but unlike in step tests it is not caused by occasional large errors but instead a smaller, systematic inability to locate the correct answer. The best algorithm only finds the solution 20% of the time, and only when the ramp slope is reduced to 4° per iteration. The algorithm is also not able to remain consistently close to the correct solution, only achieving 42 of 90 iterations when the slope was 4°. Generally the algorithm performs better when the slope of the ramp is shallower. It is assumed that the smaller changes allow the particles to gather closer to the correct answer and are more able to keep up with the changes.

As was encountered with the step tests, the PSO is overconfident in its ability to find the correct answer. In the second test in the table it is within 2° of the correct answer 17 times but reports that it has the solution 70 times, an overestimation of more than four times. The result is similar for all of the tests with ramps. As a result the PSO algorithm continues to provide information that cannot be trusted by any other components of a system.

Table 6.4: Table showing the effect of changing the slope, saved variable and distribution shape on error when tracking ramps using PSO. There were 36 iterations when the slope was ten, and 90 when the slope was four.

Distribution type	Slope	Variable saved	Average error	Reported correct	Solution correct	$\leq 2^\circ$ error
Uniform	10	SwarmBestCost	10.6	23 (64%)	4 (11%)	10 (28%)
Normal	10	SwarmBestCost	10.1	27 (75%)	2 (5.6%)	10 (28%)
Uniform	10	Bees[0].Phase1	13.1	27 (75%)	0 (0.0%)	4 (11%)
Normal	10	Bees[0].Phase1	8.19	23 (64%)	3 (8.3%)	8 (22%)
Uniform	4	SwarmBestCost	8.47	70 (78%)	8 (8.9%)	17 (19%)
Normal	4	SwarmBestCost	12.2	69 (77%)	4 (4.4%)	12 (13%)
Uniform	4	Bees[0].Phase1	5.55	75 (83%)	18 (20%)	42 (47%)
Normal	4	Bees[0].Phase1	8.10	69 (77%)	15 (17%)	32 (36%)

6.3.2.3 Ramp tests: comparison

The results for the ramp tests are similar to those from the step tests. The general trend is that the IWO produces a much better result, at the expense of testing more positions.

Most tests (nine out of 14) of the IWO algorithm produced a result that found the exact correct answer around half the time (44 to 64%, with a mean of 50%), and was within 2° almost every time (between 69 and 97% with a mean of 91%). These tests measured between 61 and 66 positions per iteration. Some variations of the IWO algorithm performed very poorly and were unable to track; these will not be compared to PSO.

Almost all of the PSO algorithm variants performed very poorly. The best result located the correct answer only 20% of the time, and this only occurred when the slope was reduced to 4° per iteration. This test was within 2° 47% of the time. For comparison, the IWO algorithm would get the exact right answer more often (typically 50% or better). Those figures also correspond to the very best version of PSO: another test did not once locate the exact correct answer, and only managed to be within 2° 11% of the time. The only redeeming feature of the PSO when tracking ramps is its small population.

IWO is relatively accurate when predicting when a solution can be trusted. When the algorithm is performing well the solution assesses the result correctly, with only two false reports. One false negative is the second iteration. The algorithm finds a solution within 2° of the correct answer, but as it is early in the algorithm's operation it incorrectly reports the solution as wrong. The algorithm also often reports that the solution is false when it is not able to find a good solution, although in this case it is typically overconfident. One test (uniform distribution, $n = 0.25$) reports 15 good answers when it in fact only found 6. In this test there were fourteen incorrect confidence levels, including both false positives and false negatives, making 40% of its reports being wrong. However, another test with poor tracking performance was only wrong in its confidence three times (8.3%).

In comparison, PSO is consistently overconfident at all times. The average amount of time spent with a good solution is 26%, but the algorithm reports that the solution is correct on average 74% of the time. One test reports the wrong confidence 20 times, or 56%. Therefore not only is the algorithm poor at tracking, it continues to overestimate its ability.

When testing the ramps, the position would change at the end of each iteration. The length of an iteration is determined (for the most part) by how many positions are tested. As IWO tests 45-65 positions per iteration, while PSO only tests ten, for a given ramp slope (quoted in degrees per iteration) the PSO algorithm is in fact tracking a faster ramp than the IWO algorithm. However, reducing the slope of the PSO ramp tests from ten to four degrees per iteration had only a small effect on the average error (albeit a larger improvement to the number of iterations with a good answer). This implies that the performance of PSO is still not comparable to that of IWO, even when the slope is similar.

To summarise, the results appear to be the same as previously: the IWO algorithm produces a good solution but tests many positions, while PSO very rapidly arrives at an incorrect answer and gives misleading reports on the quality of the solution.

6.3.3 Advanced tracking

The tests in this Section use the same test vectors (steps and ramps) as previously. However, each test has a noise signal randomly applied. In the first step test the noise signal changed only when the 'correct' signal changed. This was to simulate an environment where the noise signal is another multipath signal, in addition to the stronger 'correct' answer. In the second step test the noise changed every iteration. This was to test how resilient the algorithm is to a noisy environment and how well it can stay locked on to the strongest signal. In the ramp test the noise changed position every iteration, but the exact location of the noise (whether random or predetermined) varied depending on the test. The magnitude of the noise signal relative to the 'correct' signal also varied depending on the test.

It is not considered likely that more than one jammer will be present at once; it has been observed that incidents occur up to ten times per day [117], and typical jamming incidents have a mean duration of approximately one minute, and so are unlikely to overlap. If there are two signals present, it will be due to reflections off buildings. Research has shown that in a typical multipath environment, the strongest reflection will have a magnitude 10 dB less than that of the main signal [127]. It is hoped that blocking just the main interference signal will be sufficient to harden a receiver against jamming. Hence in this Section the algorithm will be considered successful if it identifies the strongest signal and does not get 'distracted' by the noise signal.

It would be useful for an algorithm to be able to identify multiple signal sources at once. However, the hardware (antenna and beamformer) is not currently capable of identifying multiple signal sources at once and so this will not be considered here.

6.3.3.1 Advanced tracking: Invasive Weed Optimisation

It has been demonstrated that the IWO algorithm is capable of tracking both continuous and discontinuous motion of a jammer relative to the antenna. However, in the real world there are likely to be reflections in addition to the main signal. The next question posed was if the algorithm could cope with multipath signals.

If the system was being used to locate a jammer from a moving platform it would be important to focus only on the strongest signal as this is most likely to be the true signal and not a reflection. If the system is being used for mitigation it would be useful to identify multiple angles of arrival; unfortunately with the current antenna array design this is not possible. In addition, a typical multipath environment will have reflections with a variety of magnitudes, but the strongest signal will typically be 10 dB lower in magnitude than the line-of-sight signal [127]. Hence it is hoped that only mitigating the main signal will be sufficient to prevent the receiver from being jammed.

The magnitude of the noise would vary between 0.5 and 1.0 depending on the test. This translates to the weakest signals having a magnitude that is 3 dB lower than the line of sight signal: considerably stronger than typical reflections. The assumption is that if the algorithm is able to function with a stronger (-3 dB) signal, it will definitely be able to work when multipath is a more realistic magnitude (-10 dB).

The program was modified so that two cost functions could be added together. The second

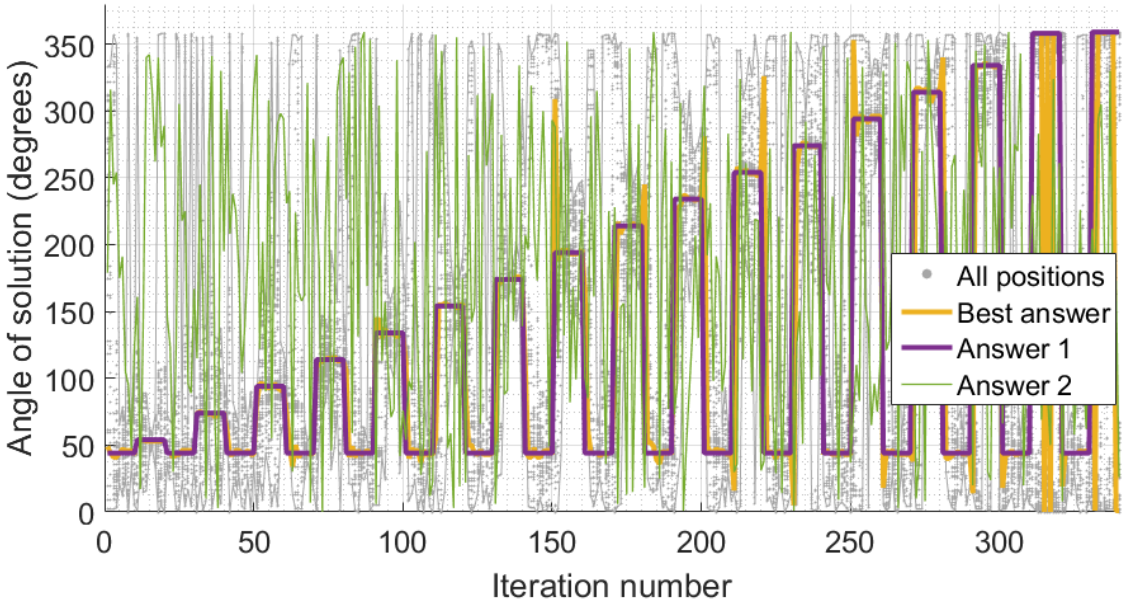


Figure 6.9: Result of using the IWO algorithm to track step changes in the presence of noise. It used a uniform distribution and $y = x$, or $n = 1$ (*cf.* Fig. 6.1). The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight). The thin green line indicates the noise for each iteration.

function could be scaled to have a different magnitude compared to the main, 'correct' signal. It is assumed that the system is linear, and therefore the two cost functions were simply added together. Tests covered both steps and ramps, with a variety of different noise scenarios added.

The first test covered was steps, with a noise signal added on top. The series of steps was the same as the previous tests without noise. The noise changed position randomly at the same time as the 'correct' signal. The second test had the noise change position each iteration, but the magnitude was kept at 0.5 as before. In all cases $n = 1$ and a uniform distribution were used.

Table 6.5 gives the results. The noise had little effect on the ability of the algorithm to locate the correct signal. When the noise changed every ten iterations the number of positions tested remained almost the same, although the error was slightly worse. The number of times the solution was exactly correct was much lower. This may be because adding a second cost function alters the shape of the minimum around the correct answer, biasing it towards the next answer over. The algorithm may find the lowest point, but data analysis does not identify this as the correct answer as it only knows the location of the stronger signal and not the effect of the weaker signal. This theory is supported by the algorithm reporting that it has found the correct answer in 241 iterations. Hence it agrees that the solution is not changing and therefore it must be in a good position.

Table 6.5: Table comparing steps for an IWO algorithm with uniform distribution when $n = 1$ with different noise added. Each run had 340 iterations.

Noise type	Positions tested	Average error	Reported correct	Solution correct	$\leq 2°$ error
None	29631	14.0	241 (71%)	242 (71%)	283 (83%)
Changes every 10 iterations	29636	14.6	241 (71%)	121 (36%)	284 (84%)
Changes every iteration	22473	10.1	62 (18%)	85 (25%)	258 (76%)

Table 6.6: Table comparing results for ramps, using the IWO algorithm with a uniform distribution when $n = 1$ with different noise signals applied. Each run had 36 iterations. Noise type numbers refer to the tests as described on page 136.

Noise type	Positions tested	Average error	Reported correct	Solution correct	$\leq 2°$ error
None	2296	0.67	34 (94%)	22 (61%)	35 (97%)
1	2098	2.14	8 (22%)	7 (19%)	30 (83%)
2	2203	6.42	6 (17%)	5 (14%)	24 (67%)
3	2069	17.3	3 (8.3%)	2 (5.6%)	13 (36%)
4	2090	1.89	22 (61%)	5 (14%)	30 (83%)
5	1943	7.28	23 (64%)	1 (2.8%)	13 (36%)
6	2006	2.33	16 (44%)	3 (8.3%)	25 (69%)

When the noise signal changes every iteration it significantly affects the distribution of positions tested. The result is shown in Fig. 6.9. The constantly changing noise will affect the depth of the minimum around the solution, making the algorithm see a changing 'best cost' between iterations and causing it to expand the range. This has an unexpected side effect: when the correct answer changes, there is more likely to be a position tested close to the new correct answer, reducing the large error during step changes and overall reducing the average error by nearly 30% compared to the no noise scenario. However, the constantly changing best cost reduces the algorithm's confidence in its answer, with it only reporting the correct answer 62 times.

The next set of tests considered the effect of noise on a ramping signal. In each test the 'correct' signal ramped at 10° per iteration. The noise type was different for each test. Full descriptions for each ramp test are as follows.

1. Noise in random positions, with half the magnitude of the main signal.
2. Noise in random positions, with a magnitude that varies randomly between 0.5 and 1.0 times the main signal.
3. Noise in random positions, with the same magnitude as the main signal.
4. Noise randomly distributed within 30° of the ramp signal, with half the magnitude of the main signal.
5. Noise 10° away from the main signal, with half the magnitude of the main signal.
6. Noise 10° away from the main signal, with a magnitude that varies randomly between 0.5 and 1.0 times the main signal

The results for these tests are summarised in Table 6.6, and compared with an equivalent test with no noise signal. It can be seen from the Table that the noise has had an impact on the performance of the algorithm. While the number of positions tested is roughly the same as without noise, the error is generally larger (and can be significantly larger, in the case of test 3). However, the fact that the algorithm is able to track the correct signal even when the noise is the same magnitude as the true signal, as in test 3, indicates that it is performing well.

Fig 6.10 shows the result of test one. The algorithm, as with the step tests, was able to track the main signal without being fooled by the multipath signal. However, as was observed before, the changing noise causes an increase in the range of positions tested. A secondary effect of the increased range is gaps between some of the positions tested, meaning sometimes the algorithm misses the correct answer by a few degrees.

The output from test 3 is shown in Fig. 6.11. While the algorithm does sometimes pick a value near the 'noise' signal, it tends to give an answer closer to the true signal more often than not. Note that this test is an unlikely scenario in the real world as two jammers of equal strength will rarely coincide. However, it is of interest to note that sometimes, particularly around iteration 23, there will be many points tested that are close to the 'correct' answer and yet a solution that is apparently further away, but close to the noise signal, is chosen.

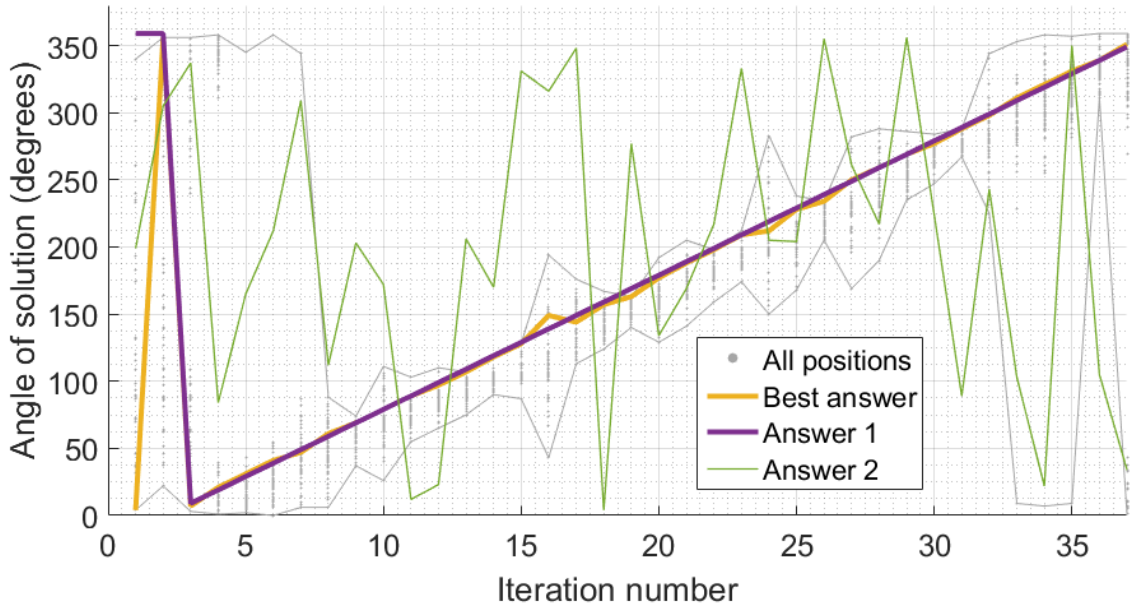


Figure 6.10: Result of using the IWO algorithm to track a ramp in the presence of noise. It used a uniform distribution and $y = x$, or $n = 1$ (*cf.* Fig. 6.1). The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight). The thin green line indicates the noise for each iteration, which had a magnitude of 0.5 and was randomly positioned.

The explanation for this is found in Fig 6.12. In iteration 23 the true signal was at an angle of 223, while the noise signal was at 52. The two cost functions are shown, along with the cost function that results from summing the two. It is anticipated that it would differ slightly in the real world due to constructive and destructive interference, but the overall shape would be very similar. As the two cost functions are almost 180° apart (171°, to be exact), they almost completely cancel out with a small minimum at around 54°. This effect would be unnoticeable if the magnitude of the noise signal was more realistic, i.e. -10 dB compared to the real signal.

Generally the algorithm underestimates the amount of time it has the correct answer. For test 1 it found the exact correct solution seven times, and thinks it was correct eight times. However, it also managed to be within 2° of the answer 30 times, but this is not reflected in the confidence of the algorithm. The average error is small (2.14° degrees) meaning that for the most part it was correct. Conversely for tests 4 and 5 the solution was only correct 5 and 1 times respectively, but the algorithm reports a correct answer 22 and 23 times respectively. These tests both had the noise signal very close to the ramp. This will have meant that the summed cost function varied less as the noise changed. The algorithm decides if a solution is correct if the cost is within 5% of the previous cost. Hence when the noise does not distort the cost function by moving large amounts, it appears to be correct more often.

Fig. 6.13 also shows a test (test 6) where the noise signal remains close to the correct answer, but with a varying magnitude (in tests 4 and 5 the magnitude was constant, at 0.5 times the correct signal). This test only finds the correct answer three times. It is overconfident, reporting it had the correct answer 16 times. However, as indicated by the low average error (2.33) the answer according to the algorithm rarely strays far from the correct

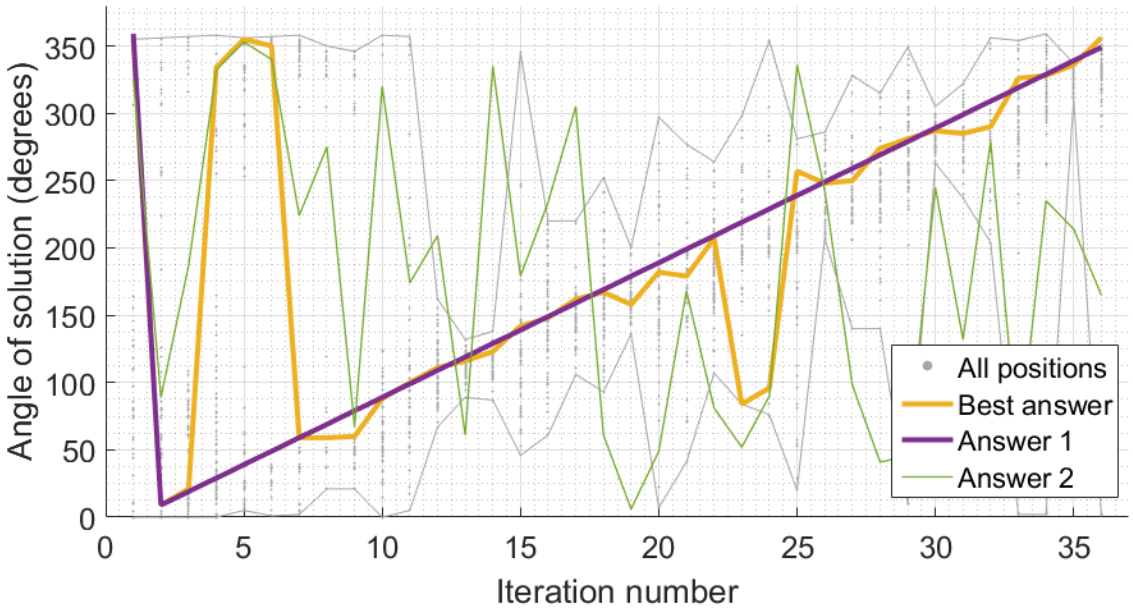


Figure 6.11: Result of using the IWO algorithm to track step changes in the presence of noise of the same magnitude as the main signal. It used a uniform distribution and $y = x$, or $n = 1$ (*cf.* Fig. 6.1). The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight). The thin green line indicates the noise for each iteration.

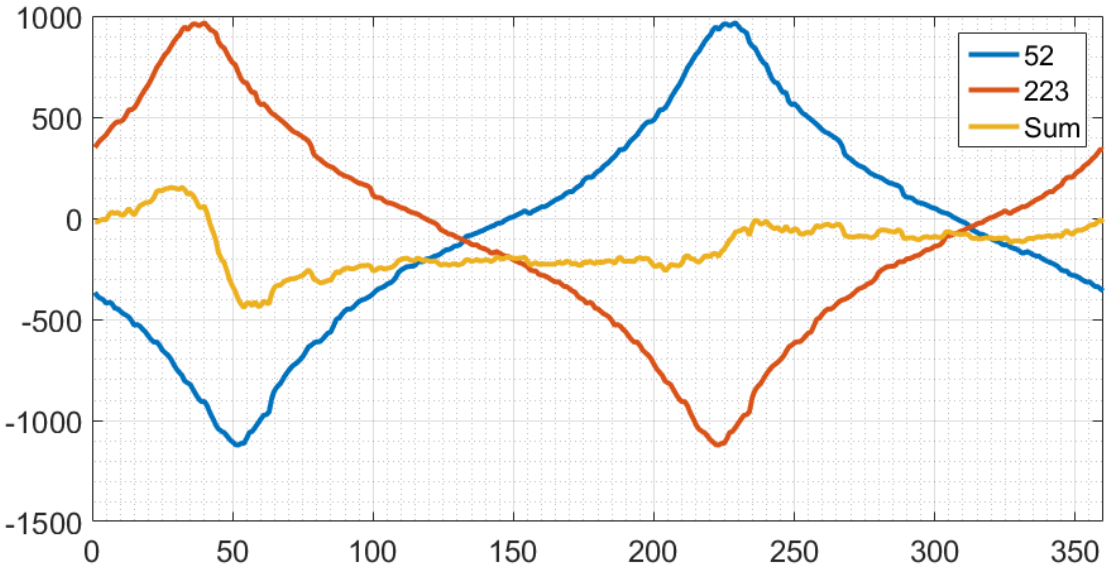


Figure 6.12: Result of adding together two cost functions that are nearly 180° out of phase.

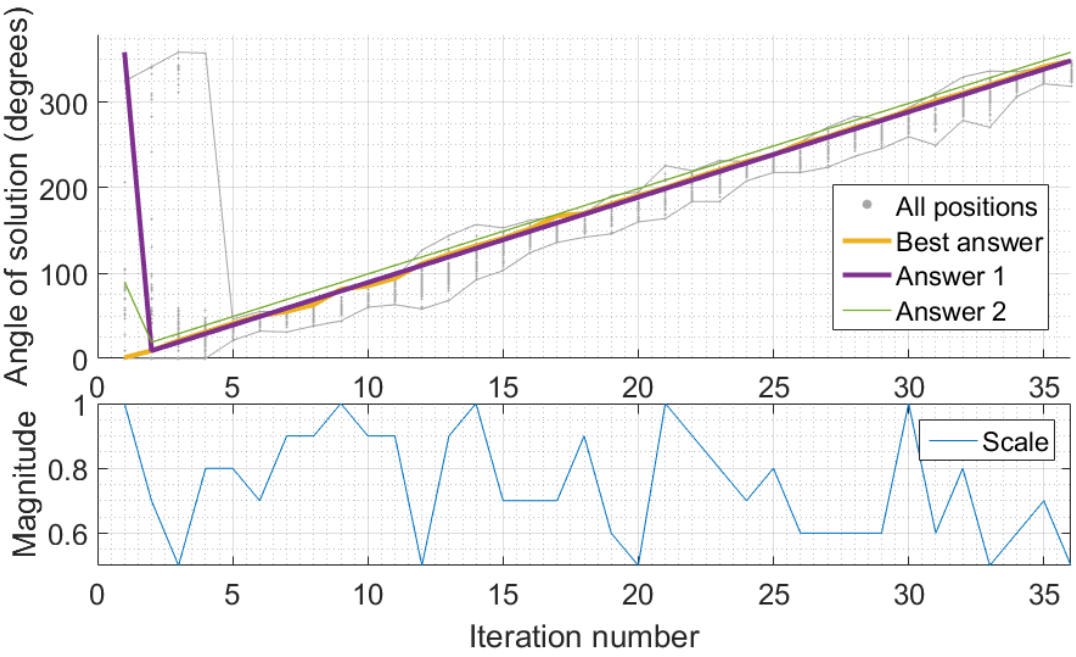


Figure 6.13: Result of using the IWO algorithm to track step changes in the presence of noise with a varying magnitude, positioned 10° away from the true answer. It used a uniform distribution and $y = x$, or $n = 1$ (*cf.* Fig. 6.1). The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight). The thin green line indicates the noise for each iteration.

answer.

The noise signal is able to manipulate the cost function and move the correct answer, as seen in Fig. 6.12. The 'correct' answer displayed on the graphs is the value for the correct answer as generated by the test program. However, the influence of the noise signal means this may be wrong, depending on the magnitude of the noise. In the scenario given in test 6, with a 10° difference between the 'correct' and noise signals, the error could be as much as 2°. Hence the error may appear to be worse than it actually is.

In both the ramp tests and also the step test when the noise changed every iteration, the number of positions tested decreases slightly compared to the no noise scenario. This is due to how the seeds are assigned to the parent weeds. When there is no additional noise signal, the parent weeds tend to be evenly distributed between the best and worst costs. When a noise signal is added the weeds are more widely distributed and the distribution of parent weed costs tends to be more heavily weighted towards the worst cost. This causes them to generally produce fewer seeds and reduce the overall number of positions tested.

One final, additional, test was run. It took advantage of the test program to provide an estimate of the resolution of the system. For this test two signals, both with the same magnitude, were initialised in the same location. It was run for ten iterations to allow the system to settle. Subsequently the two signals diverged. Each had a slope of ±2° per iteration. The results from a test are shown in Fig. 6.14. It can be seen that at first, the answer according to the algorithm sits between the two answers, indicating that it has insufficient resolution to distinguish them. However, on the tenth iteration after the split, when the separation is 36°, the algorithm selects one path and stays with it.

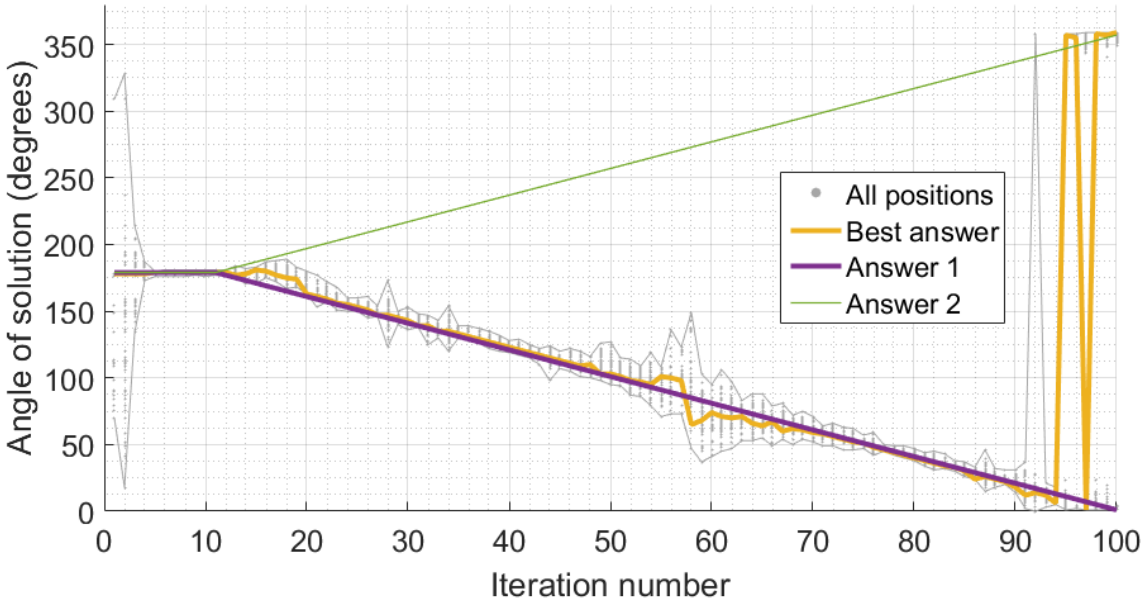


Figure 6.14: Testing the resolution with two signals of equal magnitude that diverge after a settling period. One signal is shown in purple and the other in green. The solution according to the algorithm is the yellow line. The grey dots show all the positions tested (enclosed with an envelope to highlight).

When the two positions diverge, initially the range remains small. This would indicate that the change in best cost did not change significantly. There is a general trend for the algorithm to pick values closer to the signal that ramps downwards. Inspection of the cost function shows that there is a slight distortion and it is not a perfect mirror image. This makes the lower path the more likely option for the algorithm to follow.

It is interesting to note that the algorithm only fully 'chooses' one signal over another when the separation is 36°. Test 5 had a noise signal that remained within 10° of the true signal and the algorithm was within 2° of the correct answer more than half of the time. Indeed, the errors were caused by the algorithm being unable to react quickly enough to the ramp, as opposed to being pulled away from the correct answer by the noise signal (the best cost according to the algorithm tended to be below the correct answer on the graph, while the noise signal was above). Hence, as expected, the ability of the algorithm to ignore noise signals is dependent on the magnitude of the noise relative to the true signal.

A final point of interest to notice in Fig. 6.14 is that between iterations 50 and 70 the range expands. This is the same effect as mentioned earlier that when the two signals are 180° apart the two cost functions cancel out. The rapidly changing cost at this point has increased the range.

It was observed that the number of positions tested by this algorithm was very high - it could be as many as 90 positions per iteration. While this is producing a good result, it was interesting to explore the effects of limiting the population. In the following tests the population was limited to test how it would affect the speed and accuracy of results.

Limiting the population required some modification of the code. The first change was to the variable `MaxPopulation`. After all the new weeds have been tested and the whole population sorted, the next stage is culling. Only the best x weeds are kept, where x is the value of `MaxPopulation`. The maximum population was made significantly smaller. The

Table 6.7: Table comparing results for IWO when tracking steps, using a uniform distribution when $n = 1$ (*cf.* Fig. 6.1) with different population limits applied. Each run had 340 iterations.

Population control	Positions tested	Average error	Solution correct	$\leq 2^\circ$ error
None	29631	14.0	242 (71%)	283 (83%)
20, 50	29554	14.1	236 (69%)	280 (82%)
10, 30	14041	15.1	209 (61%)	255 (75%)

second modification was larger. A part of the original algorithm [112] keeps adding new seeds until a limit has been reached. Before this point the limit had not been implemented but it was added to help control the population in the form of a variable called `MaxTotalPopulation`. If, when adding new seeds, this value was exceeded, no more seeds would be added in that iteration.

Two tests were run. The results are summarised in Table 6.7. In the 'population control' column, the first value is `MaxPopulation` and the second value is `MaxTotalPopulation`.

Table 6.7 shows that small restrictions on the population have little effect on both the speed and the accuracy. However, if the limit is more dramatic, with only ten weeds surviving between iterations and only 20 new seeds each iteration, the number of positions was more than halved. The reduction in accuracy was seven to eight percentage points with a 7% increase in the average error. Overall this demonstrates that population control can be used to tune the algorithm according to requirements, and a significant speed gain can be made from a small loss of accuracy.

6.3.3.2 Advanced tracking - Particle Swarm Optimisation

While the bees show little promise they were tested in two jammer scenarios for completeness. Tests were run, as before, using both steps and ramps with a randomly generated noise signal added. Three tests were run:

1. A ramp of 10° per iteration with a randomly generated noise signal. The noise could be located anywhere in the search space and had a magnitude half that of the 'correct' signal.

2. Steps following the same test vector as used previously. The noise changed once every ten iterations, at the same time as the correct signal. The noise could be located anywhere in the search space and had a magnitude half that of the correct signal.

3. Steps following the same test vector as used previously. The noise changed every iteration. It could be located anywhere in the search space and had a magnitude half that of the correct signal.

These tests were run twice: first with a uniform distribution, then with a normal distribution. In this case, no significant difference was observed between the two types of distribution. This is the same as was found when testing ramps or steps, as expected. They were only tested with `Bees[0].Phase1` as this method has showed better results. The population was kept at ten throughout. Test one took 36 iterations with 360 positions tested, while tests two and three were 340 iterations long and 3400 positions were tested.

Table 6.8: Table comparing results of PSO when tracking ramps and steps. The noise types are described in the text. Ramp tests used a slope of 10° per iteration and so were 36 iterations long. Step tests were 340 iterations long. All tests stored `Bees[0].Phase1`, and used a population of ten.

Test type	Average error (°)	Reported correct	Solution Correct	$\leq 2°$ error
Ramps				
No noise, Uniform	10.6	23 (64%)	4 (11%)	10 (28%)
1, Uniform	4.25	18 (50%)	2 (5.6%)	15 (42%)
No noise, Normal	10.1	27 (75%)	2 (5.6%)	10 (28%)
1, Normal	3.11	20 (56%)	5 (14%)	19 (53%)
Steps				
No noise, Uniform	23.7	211 (62%)	58 (17%)	155 (46%)
2, Uniform	53.0	194 (57%)	24 (7.1%)	93 (27%)
3, Uniform	42.3	204 (60%)	30 (8.8%)	97 (29%)
No noise, Normal	56.3	201 (59%)	33 (9.8%)	78 (23%)
2, Normal	29.7	209 (61%)	21 (6.2%)	88 (26%)
3, Normal	59.7	232 (68%)	40 (12%)	111 (33%)

The results for these tests are shown in Table 6.8. The description of the tests is given above. The no noise test results are also given, for comparison. For the ramp tests (comparing the no noise scenario to noisy tests), the noise has had no negative effect on the quality of the result. On the contrary, when a normal distribution is used, a second noise signal increased the number of times the solution was within 2° of the answer from 28% to 53%. An example result is shown in Fig. 6.15. It shows that the algorithm is still successfully remaining in exploitation mode, which is the correct behaviour with a ramped input. The algorithm manages to find the correct answer and not the weaker noise signal.

It may be that the low level noise signal (it had a magnitude of half that of the 'correct' signal) causes some jitter in the best cost, encouraging the particles to explore rather than drifting in a certain direction. The difference may also be due to random variation, although it is a large difference. All results reported here are for a single test and so further investigation would be needed. However, it is clear that adding noise to a ramp test does not significantly impair the performance of the PSO algorithm.

The noise signal has some effect on how often the algorithm reports that it is correct. For both a uniform and a normal distribution, ramp tests with noise reported fewer correct answers than ramp test without noise. This decrease, and the small increase in answers that are almost correct, means the two values are almost the same. However, during the test of a normal distribution it only assessed the accuracy well 20 times (56%). Hence the false positive and false negative rate is still high, although it does represent an improvement over the test with no noise, which only correctly assessed its accuracy 13 times (36%).

Fig. 6.16 shows the result of test two (steps, with the noise changing every ten iterations) with a uniform distribution. Overall the performance with a noise signal present was slightly worse than when it was absent, when a uniform distribution was used. It is clear from Fig. 6.16 that the algorithm is not able to switch between exploration and exploitation effectively. It explores the search space continuously in early stages, and is unable to explore in the centre portion between iterations 140 and 270. This same problem was observed with earlier tests without noise present. The algorithm's failure to change modes is reflected in the average

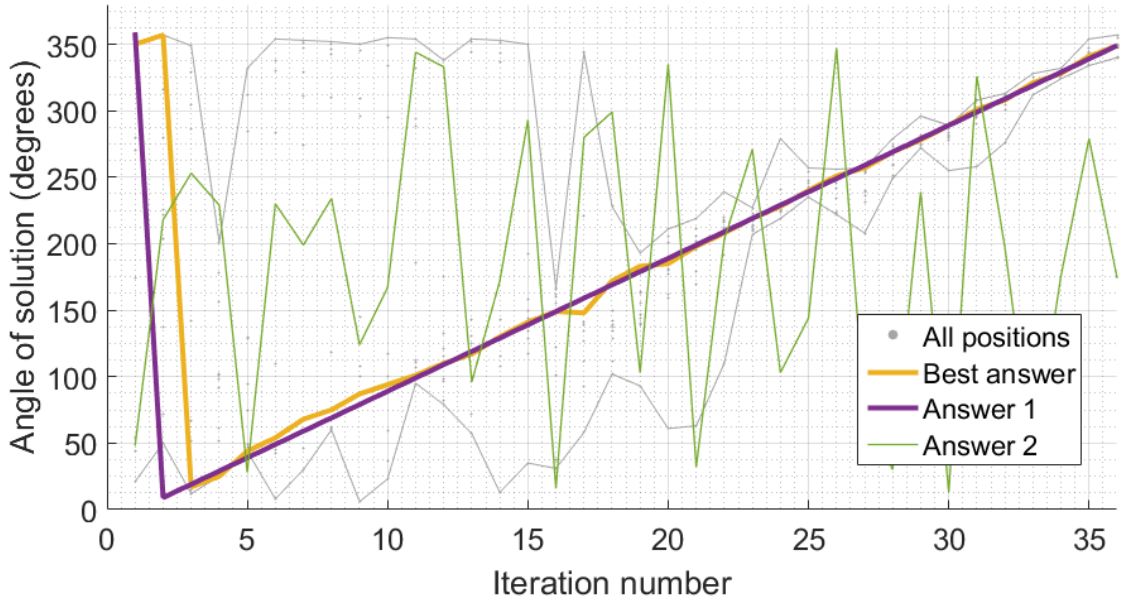


Figure 6.15: Result of using PSO to track a ramp in the presence of noise. A normal distribution was used and the ramp sloped at 10° per iteration. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight). The thin green line indicates the noise signal.

error (53.0) and the number of iterations for which the solution is correct (24 times, or 7%). As also observed before, the PSO algorithm was overconfident and predicted it had found a correct answer 194 times (57%), when the solution was in fact good only 93 times (27 %).

When test two was run using a normal distribution, the effect was slightly surprising. While the number of exactly correct answers decreased (from 33 to 21), the number of good answers increased by 3%. In addition, the average error decreased from 56.3° to 29.7°. The reason for this may be that the noise signal moving has enough influence on the objective function to cause the algorithm to move back to exploration. The difference observed between the normal and uniform distributions is less easily explained, but may be due to chance.

Test three applied to a step test a noise signal that changed every iteration. The general trend of the result was the same as previously: it degraded the performance of the PSO algorithm if a uniform distribution was used, and improved it slightly for a normal distribution. Fig. 6.17 shows the effect on the solution when a normal distribution is used. When the solution is close to the correct answer the noise does not affect the solution significantly. When the correct answer undergoes a step change, the noise signal can appear to confuse the algorithm. This is particularly obvious in iterations 200 to 220, when the correct answer is closer to the noise signal than the correct signal.

When ramps were tested with noise, the number of answers marked as correct decreased. This was not the case when the noise was added to step tests; if there was any decrease, it was small, and sometimes it increased by a larger amount (for example, adding rapidly changing noise to a normally distributed PSO test increased the number of answers marked correct from 59% to 68%). For all step tests the PSO algorithm hugely overestimated its accuracy and often reported twice as many good answers as were actually found.

Overall the PSO algorithm continues to perform poorly in the presence of noise, although

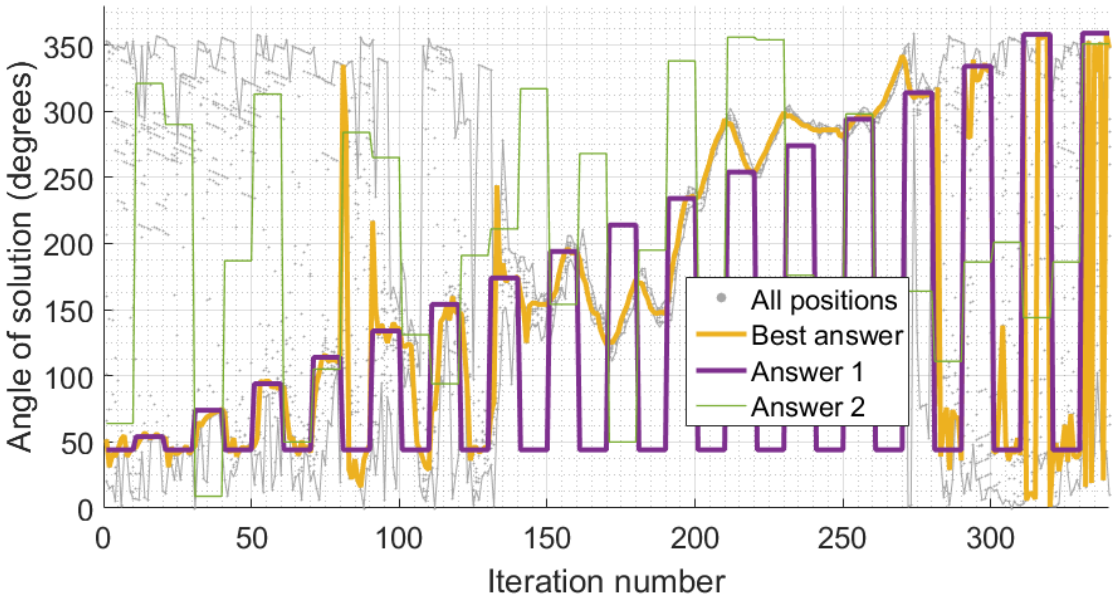


Figure 6.16: Result of using PSO to track steps in the presence of noise that changed every ten iterations. The population was ten. The noise, shown in green, had a magnitude of 0.5 and changed position at the same time as the correct answer. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

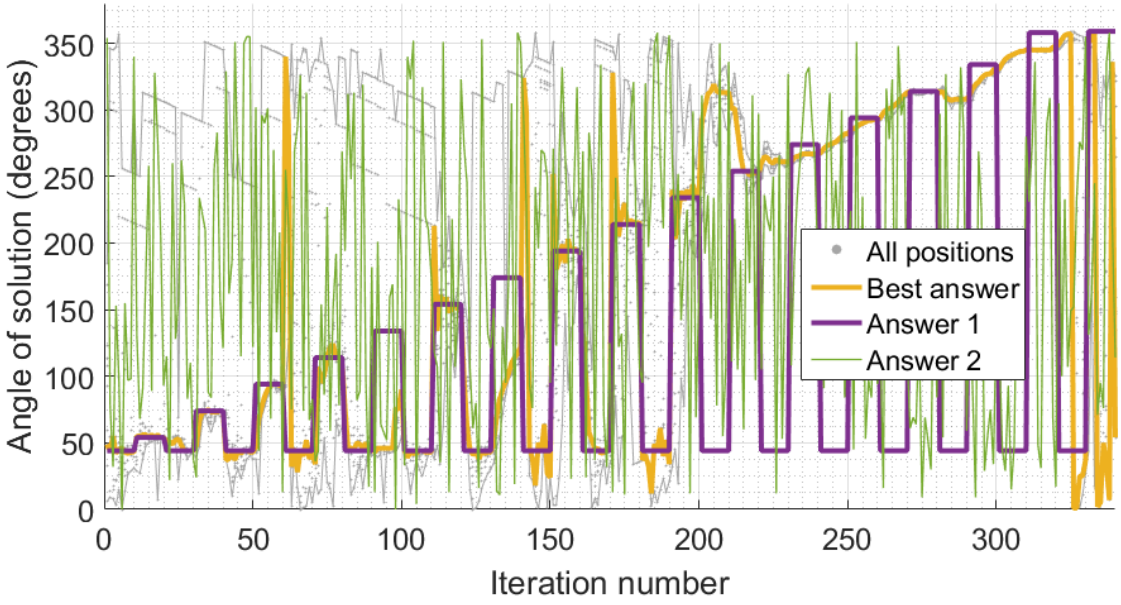


Figure 6.17: Result of using PSO with a normal distribution to track steps in the presence of noise that changed every iteration. The population was ten. The noise, shown in green, had a magnitude of 0.5. The correct answer is shown in purple. The yellow line shows the correct answer according to the algorithm, and the grey dots show all the positions tested (enclosed with an envelope to highlight).

Table 6.9: Table comparing the result achieved by the PSO algorithm compared to a random number generator. Both had a popultion of ten. The PSO algorithm does not significantly outperform the random number generator.

Test type	Average error (°)	Solution correct	$\leq 2°$ error
PSO - ramp	4.25	2 (5.6%)	15 (42%)
RNG - ramp	20.1	1 (2.8%)	4 (11%)
PSO - steps	42.3	30 (8.8%)	97 (29%)
RNG - steps	24.9	12 (3.5%)	46 (14%)

performance is no worse than without noise. It also continues to be overconfident, meaning any solution given cannot be trusted.

One final, additional test was run. It was an attempt to gauge how effective the PSO truly was. In this test the PSO algorithm was replaced with a pseudorandom number generator. This would generate ten positions per iteration (the same population as for the PSO algorithm), using a uniform distribution. The positions could be anywhere in the range 0-360 to cover the search space. This 'algorithm' was then tested against the PSO, for both ramp and step tests. The results are showed in Table 6.9.

The random number generator appears to function quite well because probability dictates that for each iteration, a position will have been generated that is somewhere near the correct answer. In fact during the step tests the average error is significantly lower for the random number generator than for the PSO algorithm, although it does not find a good error as often. The PSO algorithm has an element of the same random number generation locating a good answer by chance, particularly when the algorithm is failing to converge and instead remaining in exploration mode. However, overall the PSO algorithm does function better than a random number generator, if not by much.

6.3.3.3 Advanced tracking: comparison

This Section tested the effect of adding a second signal to the ramp and step tests shown previously. The magnitude and location of the second signal could be controlled so that different scenarios could be simulated.

The IWO algorithm performed well in the presence of noise. It generally found a good solution about as often as when no noise was present, although it found the exact correct answer less often. This reflects the fact the noise increased the range of positions tested, thus introducing more gaps between tested positions and so the algorithm may miss the exact answer more often. The average error was slightly higher, but still generally low in most cases. Rapidly changing noise also affected how often the solution was estimated to be reliable by the algorithm. It would rarely report the solution to be correct even though it was still achieving good results.

The PSO algorithm had much poorer performance than the IWO in all tests. However, adding noise did not worsen the performance significantly. Despite this, the performance of PSO is still significantly worse than IWO as it remains unable to correctly explore or exploit as required. It also remains over-confident in its solutions.

6.4 Conclusions and further work

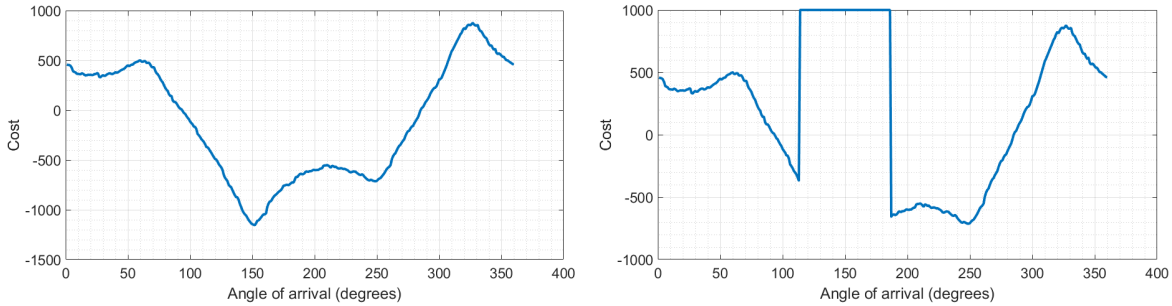
In this Chapter the optimisation algorithms created and tested in Chapter 5 were modified to be able to track a moving jammer. The two algorithms were then tested using simulations of a moving jammer that either moved constantly in one direction or made sudden step changes. To some tests an additional 'noise' signal was added to simulate multipath. While the algorithms tested in Chapter 5 had similar performance, with PSO perhaps slightly outperforming IWO, these tests of tracking algorithms showed that IWO performs significantly better than PSO. The IWO algorithm is able to locate the correct solution a significant proportion of the time, and will find a good solution (within 2°) of the correct answer almost all of the time, even in the presence of noise.

It could be argued that as PSO is only testing ten or 20 positions per iteration, it could be allowed to run for a second iteration before reporting an answer and still be at around the same speed as IWO. This assertion assumes that the part of the algorithm that requires most time is the reading of the received power. This is a reasonable assumption as typical processors run on the order of tens of MHz. The power measurements, on the other hand, are being read by a microprocessor. While the ADC is capable of 2 Msps, each power measurement is the average of 100 readings, reducing overall throughput.

If the PSO algorithm is allowed two iterations before reporting a solution, the number of positions tested would increase to 13600 (assuming a population of 20 per iteration). Restricting the IWO population reduced the number of positions tested to 14041, making the speed of the two algorithms comparable. To estimate the accuracy of PSO when given two iterations to settle, the results of a step test were taken and every other iteration discarded. This effectively gave the algorithm more time to settle on each answer. Of the 170 iterations counted, 48% (81 iterations) had an error of 2° or less. This is only marginally better than the 46% achieved by the standard algorithm. In contrast, the population-restricted IWO found a good solution 75% of the time. Hence the problem with the PSO algorithm is not the settling time of the algorithm but instead its inherent instability; the particles undergo constant motion and so have a tendency to overshoot, unlike the weeds which are static.

Overall it would appear that the PSO performs poorly no matter what conditions are applied, and so future work should only consider the IWO algorithm. The main drawback of this algorithm is its very high population, meaning it runs slowly relative to other options. Hence future work on this algorithm should focus on reducing the population. Inspection of the results files shows that when the range is small, many positions are tested multiple times. Removal of duplicates would decrease the population without any degradation of the result (as the number of different positions tested would not change). A cursory test showed that for one of the ramp tests shown in Section 6.3.1.1, if the duplicates were removed the total number of positions tested reduced from approximately 30,000 down to around 10,000 positions. Hence if a robust method for removing duplicates was created, the IWO algorithm would be a similar speed to PSO but generate much more accurate results.

It was noted that when noise signals were added, the algorithm was still able to find good results but the confidence associated with the results was much lower. In the current design the confidence is a binary value, where a 1 indicates that the algorithm believes the



(a) Before block added.

(b) After block added.

Figure 6.18: Cost function resulting from two signals at 150° and 220°, showing before and after a portion of it is made unattractive to the algorithm.

solution is correct and a 0 indicates that it thinks the solution is incorrect. The basis for the decision is if the cost of the best solution has changed by less than 5% since the previous iteration. This functions well if the best cost is always in the same range of values. However, multipath and fading effects mean that actually this value is likely to change significantly between iterations. This was particularly noticeable in step tests when the noise changed every iteration: the moving noise signal changed the value of the objective function at the best position and lead to the algorithm reporting answers as incorrect even when they were good. Future work would change this binary confidence level to a more fuzzy scale, perhaps using a different metric for if the solution was correct or not. Further experimentation would be required to find a good metric, if this confidence level was indeed required.

The current algorithm design makes no assumptions about the direction of movement of a jammer over time. When the algorithm detects a change it will search equally in either direction. A more intelligent algorithm would use information about the past trajectory of the jammer to influence its search. For instance often with the ramp tests the algorithm was only just finding the best solution as the ramp was just within its search range. If instead the algorithm recognised that the jammer was continuously moving in one direction it could extend its search in that direction without extending it in the other direction, allowing faster ramps to be tracked without searching inefficiently.

While the IWO algorithm was able to locate the strongest jammer in a given scenario, it was always limited to choosing just one solution. It was assumed that any other noise signals would be too weak to affect a receiver but this is not guaranteed in reality. Future work should focus on determining the direction of arrival of multiple signals. This might require modification of the hardware so that it can steer multiple beams simultaneously. Alternatively, on detection of a strong signal the algorithm could modify the objective function, discouraging searches in the vicinity of the strongest signal so that it can also locate a weaker peak. Knowing that the resolution of the system is approximately 36°, a section of objective function of twice 36° can be modified so that searches are discouraged. This is demonstrated in Fig. 6.18, where two solutions at 150° and 250° are summed. Once the low point of the objective function has been located (Fig. 6.18a), a portion of the objective function can be made ‘unattractive’ to the particles, encouraging it to find the next lowest point (Fig. 6.18b).

Chapter 7

Development of a real-time tracking system for moving jammers

In which: Justification for the speed upgrade is provided . . . Embedded platforms are compared . . . Considerations for FPGA design are discussed . . . An FPGA implementation is demonstrated

This Thesis has thus far presented the design of a low power, low cost GNSS jamming detection system. The system comprises an antenna array, RF beamformer, and algorithms modified to allow for tracking of moving signals. Unfortunately the current design incorporates a PC for controlling the system, something which is neither low power nor low cost. It also makes the operation of the algorithm very slow - the USB communications take up the bulk of the time taken to produce a result. This Chapter presents the work required to upgrade the platform on which the algorithm is run so that tracking of moving jammers is possible in real time.

Sweeps performed in Chapter 4 took over 13 seconds. Upon evaluating the code to understand delays and bottlenecks, it was found 98% of the execution time was spent communicating over USB. Hence eliminating the USB communications link in favour of something faster will have a significant effect on the speed even if the new platform has a slightly slower processing speed than the PC. It was decided that an embedded platform would be the best solution. A variety of types of embedded platforms covering a range of price points was considered. It was decided that an FPGA would offer the best performance for the cost and power consumption.

In collaboration with another researcher, an FPGA (Field Programmable Gate Array) implementation of the tracking Invasive Weed Optimisation algorithm presented in Chapter 6 was designed. It was synthesised for a development board (that is, for an FPGA that may not be the smallest or cheapest or lowest powered possible for the design, but instead one that was easily available for a proof of concept) and tested in the field. The tests show promise, with the algorithm able to track a moving jammer.

7.1 Motivation

In Chapter 6, both the Invasive Weed Optimisation (IWO) and Particle Swarm Optimisation (PSO) algorithms were modified so that they could continuously track a jammer. This would have the most use in a real-time tracking scenario, such as jamming mitigation at an airport. This would require the algorithm to be able to return correct solutions fast enough to keep up with the moving vehicle.

To track a moving jammer, a system needs to be able to produce solutions that are consistently close to the correct answer. It has been shown in simulations that IWO is capable of tracking a jammer moving at 10° per iteration. The next step is to establish how quickly a vehicle will move past a static receiver, so that the update rate of the system can be calculated based on this figure of 10° per iteration. It is therefore necessary to calculate how fast a car will move relative to the antenna array. Fig. 7.1 shows the set up it is assumed will be used, with a road perpendicular to the array and the closest approach being broadside to the array. This diagram shows how the rotational speed of a car can vary depending on its position: vehicle one is travelling at the same speed as vehicle two. However, the change in angle and distance between the vehicles mean that vehicle one appears to be moving significantly faster than vehicle two.

A number of different scenarios are plotted in Fig. 7.2. The distance between the receiver and the road is D, and the speed of the vehicle is S. Measurements using Google Earth show that the closest approach of roads to runways (assuming it is an airport that needs protecting, and assuming the GNSS receiver is close to the runway) can be between 150 and 300 m. A vehicle travelling at 100 km/h, 150 m away from the receiver, has a peak angular speed of 10.61° per second. Therefore if the performance of the algorithm (in terms of tracking ability on a per iteration basis) is similar to simulations, the algorithm must produce a correct answer more than once per second.

Tests were carried out to measure the communications overhead. The IWO algorithm was allowed to run and the time taken, including the time spent on communications, was recorded. This was done using Python's `datetime` library, building a cumulative sum of the time spent in certain parts of the code. Once the total time elapsed had exceeded 10 s the algorithm was stopped. The last iteration was allowed to complete; in total 13.68 s had elapsed. Of that time 13.509 s were spent on communications. This is equivalent to 98.75% of the time. This communications time includes time spent waiting for the ADC (Analog to Digital Converter) reads to be executed. As can be seen in Appendix C, a reading of the ADC works as follows: the PC issues a 'read' instruction over USB. The microcontroller reads both ADCs 100 times, then produces an average value from these readings for Σ and Δ, the sum and difference beams. When it has calculated both values, it will return them to the PC over the USB link. While this is taking place, the PC will idle and wait for the response. Because of this it is not possible to separate the time spent on communications from the time spent reading the ADCs. However, it was considered reasonable to include this in the calculation because the Python algorithm is not able to carry out other tasks while waiting for the ADC read to complete.

To make the tracking useful it must be able to operate in real time. To remove the

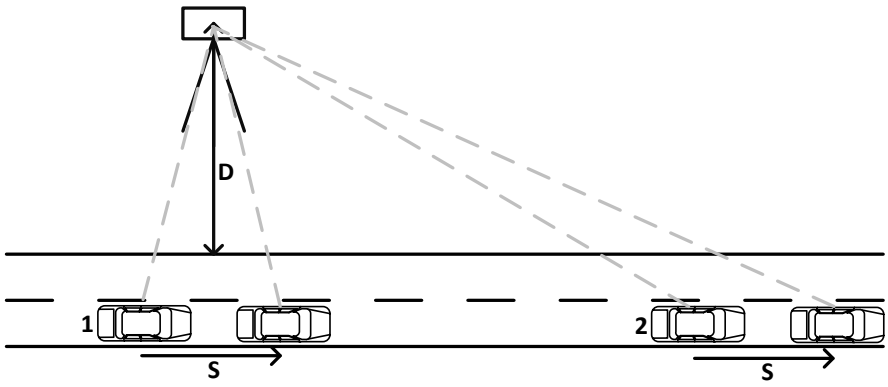


Figure 7.1: Geometry of assumed set up, with cars passing in front of the antenna array.

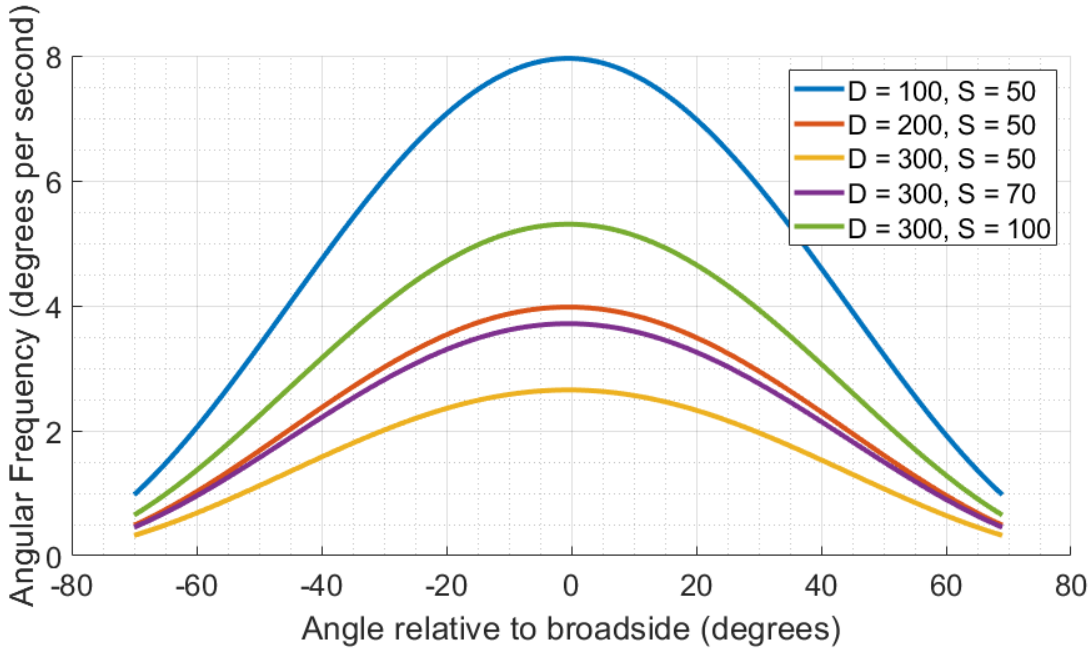


Figure 7.2: How fast cars go according to a stationary unit. D = distance between receiver and road at closest point (m), S = speed of vehicle (km/h).

communications bottleneck the algorithm must be moved from the application platform on the PC to an embedded platform that is able to control the hardware directly. This forms the basis for the case for using an embedded platform if the system is to work in real time.

7.2 Choosing an embedded platform

There are a variety of different types of embedded platforms available, each with their own features. This system is designed with two use cases in mind: a static unit based in a remote location to provide mitigation to GNSS receivers, or a UAV mounted unit for rapid localisation of jammers. Both of these use cases require a system with minimal power consumption.

The rate at which the system can produce a solution is also important. Assuming the algorithm produces the same quality of result regardless of the embedded platform, producing solutions more quickly will result in a better quality answer as the solutions can be averaged over time.

Other factors exist that will influence the decision, but not necessarily the quality of the solution. These include the ease of development for the platform, and the potential for research into novel and innovative solutions.

Cost will not be an important factor in the decision as all the platforms being considered are low cost, especially compared to a PC.

7.2.1 Microcontrollers

A microcontroller is a device that contains a CPU (Central Processing Unit) but also peripherals such as RAM and ROM. In contrast a microprocessor only contains a CPU and requires additional components to function. Small, low powered microcontrollers dominate the low cost market and microprocessors have, for the most part, moved away to other market segments. This Section will consider microcontrollers only.

There are a number of benefits to microcontrollers. They are available in very small (3 mm square [128]) packages, and have extremely low costs (on the order of £1 for a 32-bit microcontroller). Many are designed for low power applications, with manufacturers designing entire ranges of components specifically with low power consumption. Many modern microcontrollers also contain peripherals in addition to the minimum required to function: ADCs, Digital-to-Analogue Converters (DACs), and communications modules for protocols such as SPI, I^2C, UART and even USB are available. Additional modules reduce the overall materials cost as fewer components are needed, as well as making the design physically smaller. Both of these are important factors for this system.

Microcontrollers offer a simple development cycle. Code is often portable between devices with little modification, as it is written in high level languages such as C. This would make the development time required to move code from the existing Python to an embedded platform short.

Microcontrollers could be considered to be inefficient as most applications will not use every peripheral available. Hence power consumption and cost are increased by components that are not used and so it is wasted.

Microcontrollers offer little in the way of research potential. Designing an algorithm for a microcontroller presents exactly the same challenges as designing one for an applications environment (that is, on a desktop PC or laptop) as the architecture is very similar. While this will improve the development time, it also makes them less attractive for a research project.

7.2.2 Field Programmable Gate Arrays

FPGAs (Field Programmable Gate Arrays) are devices made from blocks of logic that can be programmed for a specific task. Unlike microcontrollers they do not have a specific instruction set; instead, hardware is created to carry out the instructions required. Code is written using a 'Hardware Description Language' (HDL) which describes the connections between hardware elements such as combinatorial and sequential logic, mathematical function blocks and look-up tables. The architecture means that many operations can happen in parallel. FPGAs are known for being very fast, but also for consuming significant amounts of power.

FPGAs require an external component, configuration memory, which is some form of non-volatile storage. The program is stored on the configuration memory and is loaded at runtime. New devices termed FPGA-SOCs (FPGA-System on Chips) include a microprocessor within the same silicon die. These devices are designed to minimise the number of components required and so can incorporate configuration memory, but may also include peripherals like ADCs and communications modules just like a microcontroller. This reduces the overall system size but introduces the same drawbacks as a microcontroller-based solution: namely, the power consumption and cost increase required to cover components that may or may not be used.

Despite their reputation for being power hungry, research has shown that FPGAs offer the most computing power for a given power consumption [129]. FPGAs also offer more research potential than other options; searches of the literature have found no instances of the IWO algorithm being implemented on an FPGA before.

7.2.3 Single board computers

Single Board Computers (SBCs) include devices such as Raspberry Pis and BeagleBones. They are computers where all of the required components (processor, memory and storage) are mounted on a single circuit board. They differ from microcontrollers in that they typically have more powerful processors capable of running Real Time Operating Systems (RTOSs). They are powerful enough to run bootloaders, meaning they can effectively program themselves. SBCs have the advantage that they do not require external hardware to run (reducing time spent on circuit design), but this does introduce computing overhead at runtime. SBCs may be overpowered for this application, meaning they are more expensive and power hungry than necessary.

7.2.4 Comparison

Table 7.1 lists some common embedded devices that could be used for this project. No FPGAs are quoted as their power consumption and speed are entirely dependent on the implemen-

Table 7.1: Table of example speeds, costs and power consumptions for typical microcontroller devices.

Processor	Quoted MIPS	MIPS/core /MHz	Power consumption (W)	Cost
ARM M3 (LPC1549)	125 at 100 MHz [130]	1.25	8μW/MHz [131]	£5.74 [1]
ARM A9 (TI 491NF)	7500 MIPS at 1.5 GHz [132]	2.5	1.1 W[2]	£22.39 [1]
BeagleBone Black Rev. C	2000 MIPS at 1 GHz [133]	2	1.75 W [3]	£49.11 [1]
Raspberry Pi 2	4744 MIPS at 1 GHz [134]	1.186	2.55 W [4]	£31.49 [1]

tation. The devices listed are chosen as common devices when choosing a microprocessor or SBC, which are representative of what might be chosen. A number of figures of merit are quoted in the Table. The processing power of the device is quoted in Million Instructions Per Second (MIPS), which is a measure of integer performance. MIPS measures operations such as data movement and testing of values (comparator operations). Not quoted is the more well known Floating Point Operations (FLOPs), which are a measure of a device's ability to perform mathematical operations. Many microcontrollers lack dedicated floating point hardware, but are still fully capable of carrying out the types of operations involved in the IWO algorithm, and so FLOPs may not be a 'fair' measurement.

An ARM Cortex-M3 is a very small, low cost, low power microcontroller. The particular variant chosen (an LPC1549) is manufactured by NXP and has been used elsewhere in this project. It has no floating point hardware. The ARM Cortex-M4 is an M3 with added floating point hardware, but additional power consumption too.

An ARM Cortex-A9 is an application processor likely to be found on a single board computer. They are small and low power but capable of being clocked to higher speeds (up to 1 GHz, compared to the 72 MHz maximum for an LPC1549). However, the power consumption is higher. Both the M3 and the A9 are available with built-in peripherals (making the Cortex-A9 technically a microcontroller, not a microprocessor) such as ADCs and SPI.

The power consumption and cost values quoted for the ARM Cortex-M3 and -A9 are for a single component; additional hardware for power supplies and so on would be required, increasing cost and power consumption. In contrast, the BeagleBone Black and Raspberry Pi 2 are single board computers that simply require a power source to run. Prices are liable to fluctuate with time so values are given as examples only, to allow for relative comparison between devices.

Generally, Table 7.1 shows that a more expensive processor will run faster and use more power, as is to be expected. While the SBCs operate faster, their power consumption is higher even at idle, making them less suitable for this application than a lower power microcontroller. However, there are further considerations. An SBC does not require as much hardware design time but will have more wasted power and cost from unnecessary components; a

[1] Digi-Key, 16/07/2019.
[2] Absolute maximum, assuming 1.1 V power supply. [135]
[3] Kernel idling. [136]
[4] At idle, measured using a USB power meter. [137]

microcontroller has increased design time and may end up costing as much as an SBC once PCB and component costs are included. This is especially true if development boards are used, which can be expensive.

Overall, there is little that stands out with the devices listed in the table (which does not include FPGAs) in terms of research potential. While design time may be shorter compared to using an FPGA, it has been decided that for a prototype system an FPGA is more interesting than any processor running procedural code. It may also have the best processing power given the power consumption, although further research would be required to verify it.

7.3 Existing research

The main justification for using an FPGA is its research potential. A search of the literature has uncovered no instances of the IWO algorithm being implemented on an FPGA or similar device. This is not to say that it is a bad idea: other bio-inspired algorithms have been successfully implemented. For instance, Hibbard *et al.* implemented PSO (Particle Swarm Optimisation) on an FPGA in 2013 [138]. They wished to find the optimum structure for a Bayesian Network, which is a computationally expensive process. Implementing the optimisation process on an FPGA instead of a Linux computer cluster resulted in a negligible cost saving (0.5%) but a large decrease in evaluation time (61.5% decrease). Power consumption figures were not quoted for either implementation.

Shuffled frog leaping, another bio-inspired optimisation algorithm, benefitted particularly from the parallelisation possible when using an FPGA [139]. In comparison with a 1.6 GHz Intel processor-based PC, the FPGA acheived speeds ranging from the same throughput as the PC to 2.41 times as fast. It was anticipated that the FPGA, which was only clocked at 50 MHz, would also consume less power; however, this was again not tested.

The decision to use an FPGA is also driven by the need for fast processing with the minimum possible power consumption. Research by Beasley showed that for graphics processing, an Nvidia Tegra K1 GPU (Graphical Processing Unit) processed more vertices per second than an FPGA-based implementation, but with a power consumption two orders of magnitude higher [129]. In terms of throughput per Watt, the FPGA was the best option for that particular application.

7.4 Considerations for adapting to an FPGA platform

Writing HDL code for an FPGA presents a number of challenges that are different to programming using procedural code (such as C or Python).

FPGAs differ from traditional processors in that all of the device is working at the same time, unless a part has been manually deactivated. In contrast a processor will operate on one instruction, and therefore only one or two pieces of data, at a time. (Threading is an exception to this, but as most microcontrollers are not capable of threading it will not be considered here.) The result is that an FPGA is capable of performing much faster, but timing requirements are much more stringent. Data may be produced by one block and used

by another as soon as it is ready, meaning the clock and data must be synchronised across the device.

Another key difference is the design decisions that can be made. When writing code for a processor, code is compiled so that it uses a combination of the operations that are possible on the target device. The available operations are restricted by the hardware. In contrast, FPGAs offer more flexibility. A certain task could be carried out in the same way as on a microprocessor if the resources are available; alternatively much smaller, faster options are available if the accuracy of the answer is less important.

Complicated operations on FPGAs are often run using IP (Intellectual Property) blocks. These are designs created by vendors and licensed to users. They appear as 'black boxes', operating in a defined way but with the exact implementation hidden. These decrease the time taken to implement the design, but restrict the design to one manufacturer. As this is not ideal at the early prototyping stage, no IP blocks will be used in this design.

Random number generation

The original algorithm proposed in [112] uses a normal distribution when generating new positions. However, when the algorithm was modified to allow tracking of moving emitters, it was found that a uniform distribution slightly outperformed a normal distribution. In the Python implementation the `random.randint()` function was used for the uniform distribution and `random.normalvariate()` was used for the normal distribution. By a quirk of the `random` library's programming, `random.randint()` is actually slower than `random.normalvariate()`. This is not usually the case as generating a normally distributed random number typically starts with a uniformly distributed random number, before normal operations are performed. A review [140] notes four distinct approaches for implementing a normally distributed RNG (Random Number Generator), all starting with a uniform RNG and then optimising for parameters such as speed or size. Ultimately, in this case, speed is not an important consideration as the time spent on communications is higher than the time spent on calculations. What is more important is the quality of the RNG. Quality in this case refers to the level of randomness. The Python `random` library is *pseudo-random*, which means that it is not truly random. Numbers are produced in a predictable sequence that is very long (in the case of the Python `random` library, which uses a Mersenne Twister [141], the repeat is $2^{19937} - 1$ digits long). The result will also not produce the same value twice in a row. For most applications, this level of randomness is sufficient.

When implementing a RNG in HDL, the standard approach is to use a Linear Feedback Shift Register (LFSR). An example LFSR is shown in Fig. 7.3. An LFSR is a shift register with a number of 'tap points'. The example shown is eight bits long and has three tap points at bits 1, 2 and 3. The values of the tap points are exclusive-OR'ed together. The result of this operation is inserted into the left-most bit of the register and the other values are shifted right by one. The generated random number is the value of the whole register.

When implementing an LFSR there are a number of factors to consider. The length of the sequence is determined by the size of the register but also the tap points. Some combinations of tap points will produce a sequence the length of the largest number that can be represented by the register (that is, an 8-bit register can create a 255-bit sequence). These are known as 'maximal length' and are the optimum design for the register. The sequence produced by an

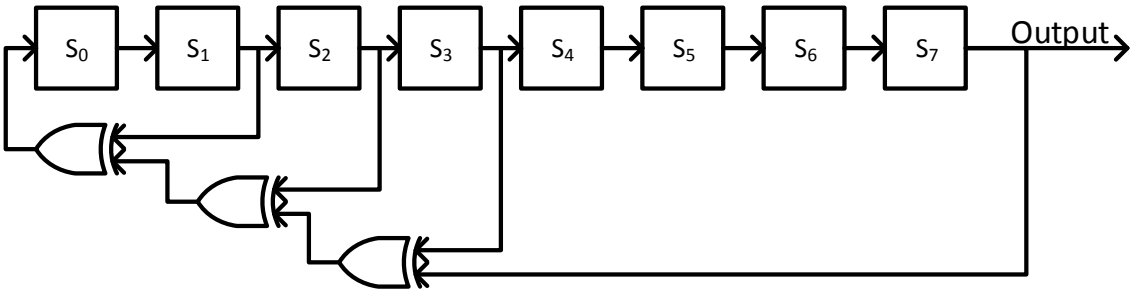


Figure 7.3: An 8-bit linear feedback shift register with tap points at bits 1, 2 and 3.

LFSR will seem random but is in fact a predictable sequence. They will also never produce zero. Not including zero, however, the distribution appears uniform.

An LFSR has a significantly shorter sequence length than the Mersenne Twister used by the Python library, but the implementation takes up much less space. An 8-bit LFSR is one register and two gates, while a Mersenne Twister uses 2.5 KiB of buffer, in addition to any space required for logic. The Mersenne Twister is also capable of producing zero, although in this case it would not be needed so this is not a consideration.

Ultimately, bearing in mind the goal of making this system as small and fast as possible, it was decided to implement an 8-bit LFSR as the RNG. Its small size and fast operation are ideal for this project. It also produces a uniform distribution, which performed better than a normal distribution in tracking simulations.

A problem arises with the exact implementation of the IWO algorithm. The range in which the random numbers are required changes with every iteration and every parent weed. As such the output of the LFSR needs to be restricted to the required range. Two initial methods were proposed to deal with this, each with tradeoffs. The first is not predictable in the time required to produce a value, but the distribution is uniform; The second will produce a value within a predictable amount of time but the distribution is not uniform.

The first method is to simply discard generated values until one within the desired range is generated. As the LFSR is synchronous with the system clock and one number is generated per clock cycle, and a large number of numbers may be discarded, this method could be potentially slow.

The second method attempts to apply an additional layer of randomness to the process. The number is taken and the length of the allowable range is subtracted repeatedly until the value of the generated number is less than the allowable range length. The generated number is then added to the lowest value of the range. This method biases the distribution function to the lower end of the range so that the distribution is not uniform.

Upon consideration, the first of these two methods was chosen. It was decided that a biased distribution would reduce the quality of the answer and it was preferable to increase the amount of time taken. This additional time should not increase the overall time required for the algorithm to run as random number generation is not the bottleneck and can take place in parallel with other, slower operations.

Number representations

In the IWO algorithm, there are times when more precision is required than integers can provide. Hence non-integer numbers are required. There are two approaches for representing

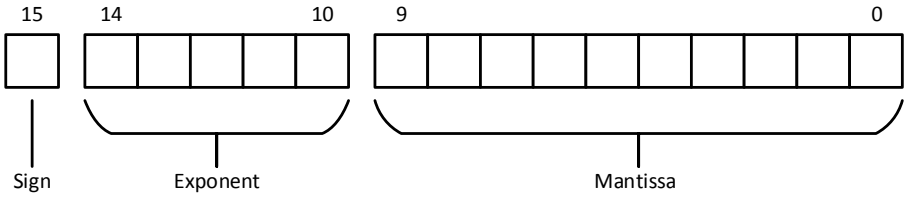


Figure 7.4: An example 16 bit floating point number, with sign, exponent and mantissa components.

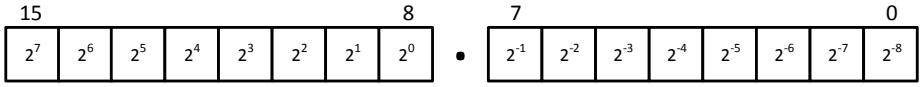


Figure 7.5: A 16 bit fixed point representation. The position of the binary point (here, between bits 8 and 7) would depend on the implementation.

non-integer numbers in hardware: floating point and fixed point. In floating point the number is split into three parts - the sign, the mantissa and the exponent (see Fig. 7.4). The number being represented is $-1^S \times (M \times 2^E)$, where S is the sign, M is the mantissa and E is the exponent. This representation allows for large numbers and very small numbers to be displayed without needing as many bits as if fixed point is used. It also allows for an increase in precision as fewer leading or trailing zeroes are needed.

A fixed point representation is similar to how integers are represented, but the less significant bits represent fractional numbers instead of integers. This method has much less range and precision than floating point but is smaller and easier to handle in hardware. Both number systems were tested and it was decided that when implementing this algorithm, only fixed point representations would be used. The range of numbers needing to be represented in this algorithm is small, and the loss of precision appears to have little effect.

Removing higher powers

The IWO algorithm requires numbers to be raised to a power during the calculation of the range. It was established in Chapter 5 that $n = 3$ produced good results, and was chosen as it was recommended by the literature. This was carried out using the standard operator for powers in Python, namely $x**3$. HDLs such as VHDL and System Verilog do not include a power operation as standard and it is up to the designer to implement the function.

Raising numbers to arbitrary powers is a complicated calculation. To simplify it, the arbitrary power y^x is first simplified using (7.1).

$$y^x = \exp(x \log y) \tag{7.1}$$

This version allows approximations for exponential and logarithmic operations to be used. Typically these use the Taylor-Maclaurin expansion to estimate a solution. A low-latency solution has a latency of 23 clock cycles [142] and requires a number of look-up tables, meaning the resource use is large.

The algorithm being implemented in this research requires a cubing operation. Rather than implement a block to raise a number to any power, flexibility was traded for size and instead it was calculated manually. That is, a^3 becomes $a \times a \times a$. As this can be implemented

using fixed point logic and requires no lookup tables resource use is small and it can be calculated in one clock cycle using a DSP (Digital Signal Processing) block within the FPGA. Additionally, floating point numbers are still not required, reducing complexity further.

Identifying parallelisation

The main difference between an FPGA and a processor (normal or micro) is that a processor will operate on one instruction at a time. While innovations such as pipelining and threading have somewhat blurred this distinction, the fact remains that operations are carried out in order, with only a small number occurring at once even in the most parallelised cases. Conversely, an FPGA will carry out all operations simultaneously, barring program flow control methods such as state diagrams and enable signals. The algorithm as written in Python is designed to run sequentially, which may not be the most time-efficient method. Hence the algorithm should be modified when ported to an FPGA to take advantage of parallelisation where possible.

There are several points in the code where parallelisation could be used. For example, in the implementation discussed in Chapters 5 and 6, the algorithm will give the hardware new phase shifter settings, then request a reading of Σ and Δ. It will then wait until the ARM microcontroller replies with the ADC values, which takes some time. By using parallelisation, this delay may be reduced: while the FPGA is communicating with ADCs it can also be calculating new seeds, for example.

Another approach to parallelisation is to consider when the same operation needs to be carried out multiple times. For instance, random numbers need to be generated when creating new seeds. One piece of hardware could be instantiated and every random number generated serially by this one block. Alternatively, at the cost of additional power and resource use, many instances of random number generators could be created. This would allow more than one number to be generated at any one time. It would only be worth duplicating hardware that is found to cause bottlenecks. As it is expected that the slowest part of the code will be the reading of Σ and Δ, in this case duplication of hardware is unlikely to be applicable.

7.5 Results

The HDL code discussed in this Chapter was written by Dr. Alex Beasley based on the diagrams given in Appendix C; therefore it will not be described in detail. It is the same tracking IWO algorithm as was discussed in Chapter 6, with the modifications for HDL discussed above. While the code is not discussed, the results of simulations and field tests are presented here.

The new, FPGA-based, algorithm was first tested in simulations using ModelSim [143], an FPGA/ASIC simulation tool. It was subsequently used to program an FPGA on a development board and was tested in the field using a real jammer.

7.5.1 Simulation results

The algorithm was evaluated using a number of different tests. The results are shown in Table 7.2, alongside some comparable results from the Python-based tracking IWO algorithm from Chapter 6. For the comparison the final column was changed slightly: for the Python

Table 7.2: Table comparing results of simulations by both the Python script and the HDL program.

Test	Platform	Average error (°)	Solution correct	Error $\leq$ 2 steps
10 degree ramp 1	Python	0.67	22 (61%)	35 (97%)
10 degree ramp 1	HDL	104.7	0 (0%)	0 (0%)
10 degree ramp 2	HDL	35.13	0 (0%)	1 (1.4%)
4 degree ramp	HDL	5.59	9 (10%)	38/90
Increments (10°, no spikes)	HDL	5.65	8 (4.5%)	81 (46%)
Increments (10°, spikes)	HDL	27.4	6 (3.5%)	39 (22.7%)
Increments (4°, no spikes)	HDL	4.34	16/151	82/151
Step tests	Python	-	242 (71%)	283 (83%)
Step tests	HDL	-	10 (3%)	193 (58%)

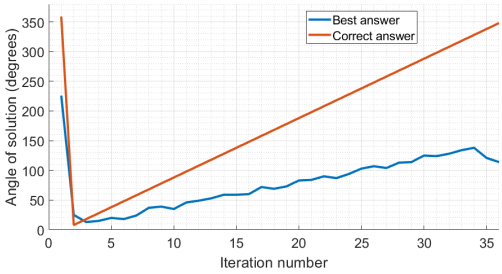
simulations there were 360 positions ranging from zero to 359. The HDL simulations still had a range of zero to 359 but in steps of approximately 1.4° (*i.e.* 360/256, the step size for the phase shifters used). To make the results comparable the final column of the table was modified so that it records when the error is within two steps, rather than when it is within 2°. A step for the simulated Python is 1°, while a step for the HDL code is roughly 1.4° due to the conversion from 256 positions over 360°.

Three types of tests were run. The first test was a ramp. As with the Python-based algorithm, the correct solution would start at 0° and increase by a fixed amount each iteration. When the ramp slope was 10° the HDL-based algorithm was unable to keep up with the slope (ramp 1, Fig. 7.6a). The test was then modified so that the change in solution happened every other iteration (ramp 2, Fig. 7.6b). The algorithm performed better, with a lower average error (reduced from 104.7° to 35.13°), meaning it was closer to correct. However, it was still unable to keep up with the ramp. However, when the slope was decreased to 4° per iteration, as in Fig. 7.6c, the algorithm is able to track successfully. The average error remains higher than for the Python implementation (5.59° compared to 0.64°). This average error is more comparable to the PSO algorithm.

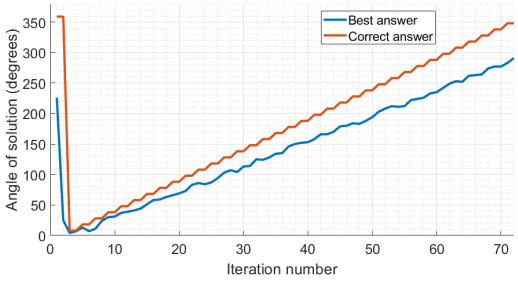
The second type of test was increments. In this test the position changed whenever the algorithm declared that the solution had been found. Increments of both 4 and 10° were tested. The algorithm typically took three iterations to settle on a solution, which is what was seen with the Python-based implementation.

The initial version of the code had a reset condition that would trigger if the correct solution changed by a large amount; however, it was found to be oversensitive. The effect of this is visible when comparing the two tests with 10° increments. Fig. 7.7a shows the result with the reset condition while Fig. 7.7b is without. When the reset condition is in place, the algorithm will regularly reset the system and produce a best answer at 226 (this value represents an artefact of initialisation). With the reset condition removed as in Fig 7.7b, the algorithm continues to track instead of resetting. As a result it produces a good answer more often (although it is correct approximately the same proportion of the time). Note that for the first iteration of the test in Fig. 7.7b the solution is 226, as this solution is reported immediately after a reset (when the system starts).

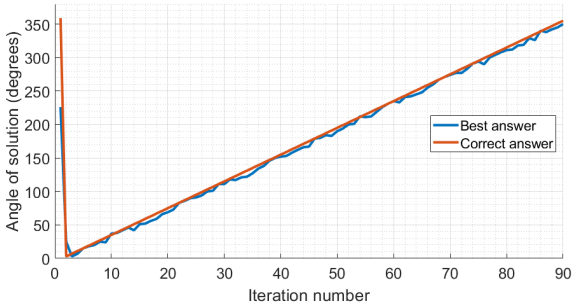
The final test was increasing steps, using the same test vector as in Chapter 6. Fig. 7.8 shows the results of a step test. Once again the spikes are present, showing that the algorithm



(a) Ramp of 10° per iteration. Algorithm unable to track.

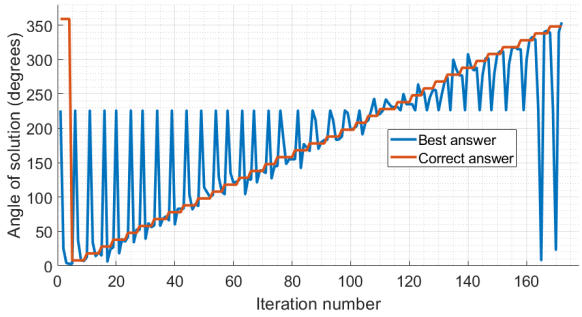


(b) Ramp of 10° every other iteration. Algorithm still unable to track, but closer

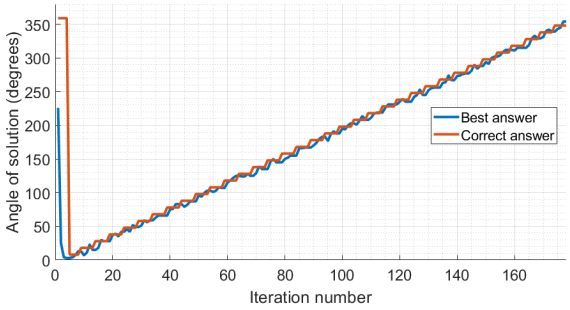


(c) Ramp of 4° per iteration. Algorithm is able to track successfully.

Figure 7.6: Ramps of different slopes to test the limit of what the HDL-based algorithm is able to track.



(a) Ramp test before fixing the error.



(b) Ramp test after fixing the error.

Figure 7.7: Two ramps, before and after the reset error was found and fixed, showing the effect of the error compared to the intended behaviour.

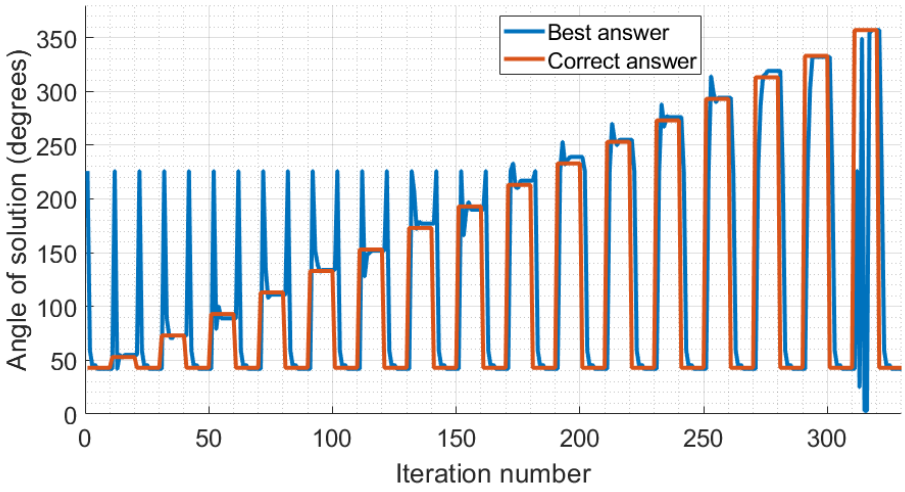


Figure 7.8: Step tests for the HDL-based algorithm, with the reset error still present.

is reacting (or possibly overreacting) to changes in the solution. While the tracking of slopes was comparable to the performance of the PSO algorithm, when tracking steps the HDL-based IWO algorithm significantly outperformed PSO. Fig. 7.8 shows the algorithm behaving correctly, exploring when it recognises the solution has changed and exploiting when the solution is static. As with the Python implementation the solution wraps, as evidenced by the solutions between iterations 310 and 320. The correct solution was at 359°, and the algorithm sometimes produces solutions both around 359 and around 0°. The solution produced by the HDL algorithm is generally worse than the Python-based version, but it is managing to track close to the correct answer.

Simulations do not take into account the fit of the design onto the FPGA and so do not have any restrictions on the clock speed. All simulations were run with a clock frequency of 50 MHz. At this speed each iteration took approximately 620 μs. During that time it spent 250 μs on calculations. However, due to parallelisation, it is not possible to say that the communications took the other 370 μs. What is reasonable to assert is that the communications is the bottleneck, and the time taken for one iteration is not dictated by the calculations portion of the program. The iteration time takes into account the speed of communications with the external ADCs and so is realistic. The time taken for one iteration is dependent on the number of positions tested so this number is only approximate.

7.5.2 Fitting the program to FPGA hardware

The HDL code has been verified in simulation and shown to be a little worse than a Python implementation, but capable of tracking. The next step is to configure it for a real device so that resource use can be assessed.

The fitter returns a number of key figures regarding resource use on each device. At this time only Intel devides were considered so that they can be compared more easily. However, the HDL written has no IP cores and so is generic and could be ported between different manufacturers' devices.

ALMs needed: ALM is an intel-specific term, which stands for Adaptive Logic Module. They are blocks within the FPGA for creating basic functions such as combinatorial and

sequential logic, as well as simple addition and subtraction. The proportion of the device used is given in brackets in the table.

ALUTs used: ALUT is another Intel-specific term, meaning Adaptive Look-up Table (LUT). LUTs are a key part of FPGAs, allowing results of common operations to be stored and looked up instead of calculated at run-time. The 'adaptive' part simply means that an 8-input ALUT could also be configured as, for example, two 4-input LUTs. This saves resources compared to having to partially use two seperate 8-input LUTs.

DLRs used: DLRs are Dedicated Logic Registers. This is just the term used in the fitter report to describe a flip-flop; that is, the number of one-bit registers required for the design.

DSPs used: DSPs are Digital Signal Processing blocks. They are effectively multiply-accumulate blocks, which is a common operation in signal processing. They are also optimised for floating point mathematical operations.

85° restricted $\mathbf{F_{max}}$: This is the maximum frequency at which the device can run at a silicon temperature of 85°. It is slightly slower than if the device was kept cooler due to how silicon reacts to temperature. However, this value gives a good indication of the maximum speed at which it is safe to clock the device.

Total power consumption: This is the estimated power consumption of the device. The dynamic power of the device is also quoted. Static power is that dissipated by leakage through gates and is dependent on the device, not the configuration; dynamic power is the power dissipated by switching transistors, and is the difference between the total and the static power. The rest of the power is used for powering the I/O (Input/Output) banks and in switching, which is a lossy process. Some of the devices listed in Table 7.3 are FPGA-SoCs, which means a microprocessor core is embedded in the same silicon. The microprocessor is deactivated for this design so consumes no power.

The process of fitting a design onto an FPGA begins with a random seed. Some blocks are placed randomly according to this seed, and the rest of the design is built around this starting point. As a result some results are better than others and so an average is taken. In the data used to produce Table 7.3 the range of values was very small, indicating that the quality of the fit does not vary significantly with the seed. As a result statistics regarding the mean and standard deviation of results are not given as they provide no information of interest. The results in the table are the average over ten fit runs.

Table 7.3 gives fitter results for four different devices. The Cyclone V is a low cost device. It has been largely superseded by the Cyclone 10 series but is included for comparison. The Cyclone 10 LP is a device optimised to have as low power consumption as possible. The Cyclone 10 GX is a faster, higher performance device optimised for applications with large amounts of communication, as it has many dedicated transceiver blocks. Finally the Max 10 is a very low cost, low power device. Max devices generally have fewer areas of specialised logic (such as floating point functions, transceivers and so on) compared to Cyclone devices.

A quirk of the Cyclone 10 LP and Max 10 devices is that, in order to make the power consumption as low as possible, for every ALM there must be one ALUT used. Hence the usage of both block types is always the same for those devices.

The lowest power device is the Cyclone 10 LP. This device also has the slowest clock, but it is still faster than that used in simulation. As the simulation was able to track, this

Table 7.3: Fitter results, showing resource use for various FPGAs.

Device	ALMs needed	ALUTs used	DLRs used	DSP blocks used	85° restricted F_{max} (MHz)	Total power consumption (Dynamic power) (mW)
Cyclone V 5CSXFC6D6F31C6	11403 (27%)	15012	21303	5	101.37	419.77 (8.55)
Cyclone 10 LP CL120ZFF780I8G	29431 (25%)	29431	19788	4	75.96	110.25 (39.87)
Cyclone 10 GX 10CX220YF780E5G	10826 (13%)	12882	21141	3	180.09	2085.24 (1407.43)
Max 10 10M50SCE144I7G	29555 (60%)	29555	19788	1	84.97	249.65 (8.58)

should be adequate. The Cyclone 10 GX has by far the highest power consumption, which is not surprising as this device is designed for speed rather than low power consumption. It is capable of a clock speed almost twice as fast as the second fastest device (the Cyclone V, which is also not designed for low power consumption).

In terms of resource use, the Cyclone V, 10 LP and 10 GX all had only a small proportion of the ALMs being used. This would imply that in the final design, a smaller (and therefore lower cost) device from the same family could be used. However, it was not considered relevant to find out exactly what the smallest possible device would be at this stage of the design process.

It is difficult to compare the power per operation between an FPGA and a microcontroller. A microcontroller carries out operations serially, and so the number of operations per second is fixed for a given device. In contrast the hardware in an FPGA is application specific. That is, the number of operations per second will depend entirely on the implementation, as a highly parallel design will naturally have more operations carried out per second than a design with just one pipeline. Table 7.4 has some approximate values for power per MIPS for the devices already mentioned in this Chapter, compared to one FPGA. Note that the figure for MIPS vs. frequency given in Table 7.1 is for the processor core alone. The number cannot be compared to other devices as the low power core could not be clocked high enough to achieve comparable throughput, and connecting many microcontrollers in parallel to increase throughput would add so much overhead that there would be little gain in speed. The figure given for an M3 in Table 7.4 is for a real device, the LPC1549 from NXP [104]. It includes the peripherals required for the microcontroller's operation.

The value for the Intel Cyclone V was a headline figure given on the Intel website [144]. It applies to a fully utilised FPGA. In this case it was found that the power consumption was only 419 mW (compared to 1.8 W maximum) and so it can be assumed the MIPS figure is lower. Knowing that it takes approximately 250 μs to generate an answer, it can be assumed that it takes 233,000 operations for one iteration (not including parallelisation. Taking into account the different clock speeds of the different FPGAs and their different power consumptions, and based on these assumptions, the best power per MIPS figure for the chosen sample of FPGAs was for an Intel Cyclone 10 LP, which achieved 6354 MIPS.

Table 7.4: Power per MIPS figures for microcontrollers, compared to a Cyclone V FPGA.

Device	MIPS per milliwatt
ARM M3	3409
ARM A9	6818
Beaglebone Black	1143
Raspberry Pi 2	1860
Intel Cyclone V	2222
Intel Cyclone 10 LP	6354

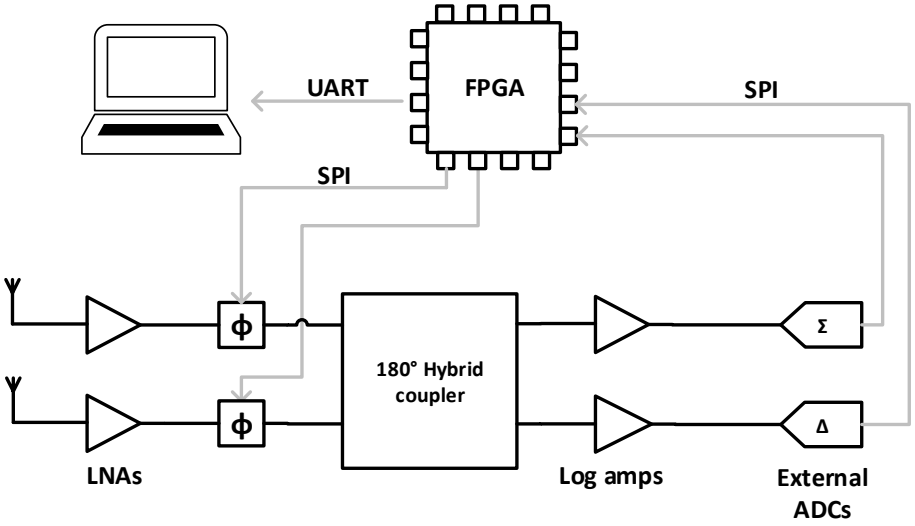


Figure 7.9: The setup of the hardware for field tests of the FPGA-based system.

This is comparable to the ARM A9. Not included in these figures is the throughput (MIPS) per Watt per time, as this was not possible to calculate for any of the microprocessors as the amount of parallelisation is unknown. However, these numbers show that an FPGA, even without significant time spent on optimisation, can achieve a MIPS per Watt value in excess of many common, low power processors.

7.5.3 Field tests

The final results presented in this Chapter are from field tests of the complete system. In order to keep development time as short as possible, a DE1-SoC development board from Terasic was used. This board hosts a Cyclone V FPGA, as well as all the peripherals required for a complete FPGA system. As it is a development board there are also devices such as transceivers, LEDs and other miscellaneous parts that may be desired by a developer. As a result the overall power consumption of the board is very high.

The system was set up as shown in Fig. 7.9. A two element, one dimensional search was used as before. The laptop is still used but only as a means of collecting data. The FPGA runs the algorithm, controlling the phase shifters and reading the values from ADCs. The FPGA has no on-board ADCs and so external ICs were used. At the end of each iteration the FPGA would report back to the PC the best position, and the cost at that position. In turn, the PC had a Python script that would read this data and store it, along with a timestamp. The FPGA reports data to the PC constantly, but the PC will only record this data when the Python script is running.

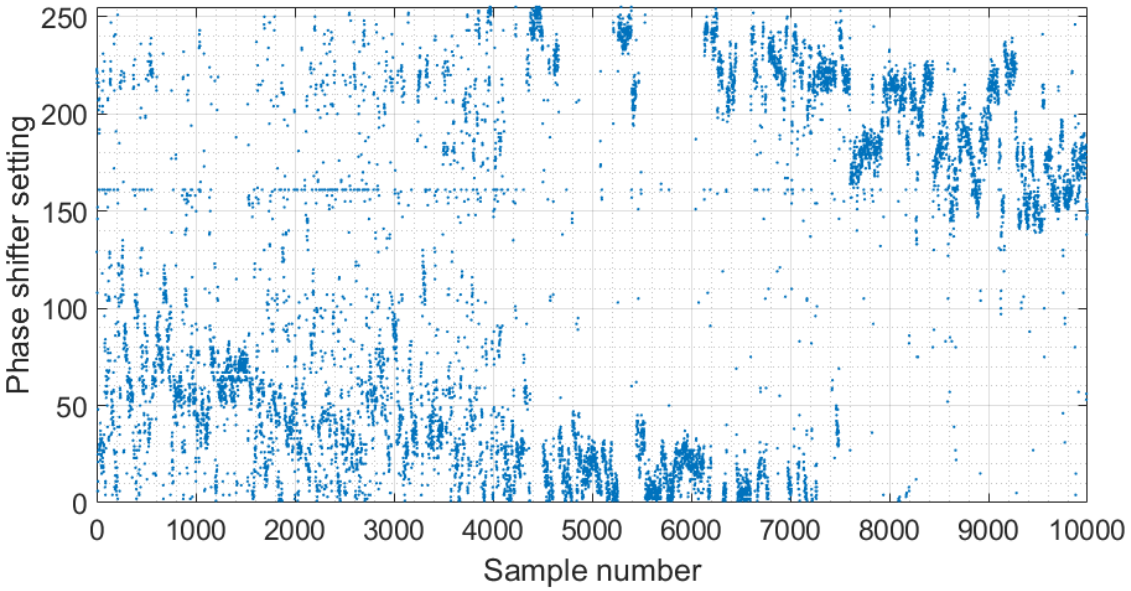


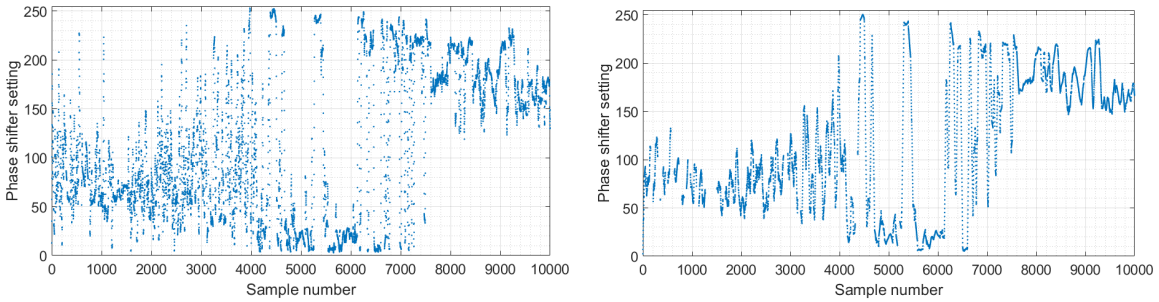
Figure 7.10: Raw measurements (with no averaging) showing the jammer moving at walking pace from left to right.

To test the HDL-based algorithm, the Python script on the PC was instructed to record 10,000 samples. The jammer was then carried, at walking speed, left to right. Fig. 7.10 shows the results of one test. The graph shows the phase shifter setting of the best result (in the range of 0-255), plus the sample number. The sample number was used rather than the time because the Python script had a tendency to give the same timestamp to a block of results. The graph shows that over time, the reported position moves. The solution has a great deal of noise, and the problem with the spikes (caused by the over-sensitive reset, which had not been diagnosed at the time of testing) is present, manifesting as a line of points at 161. However, there is a general movement trend visible.

There is a large step in the solution that occurs between samples 4000 and 7000. This is due to the solution wrapping, meaning that a solution at 0 is adjacent to a solution at 255. 'Unwrapping' the solution would be possible but only raw data will be presented here.

According to the timestamps, the system creates a solution approximately every 1.5 ms. This is slower than the simulation predicted, but more than 1000 times faster than the Python-based system. At this speed averaging can be applied without a significant performance decrease. Fig. 7.11 shows the results of applying a 10-point or 50-point moving average to the results. The 50-point moving average produces a much smoother line. The update rate is not affected when a moving average is used.

There are a number of reasons why the result is noisy. In Fig. 7.10 two clear sections to the data can be seen. The first ∼4000 samples are significantly more noisy than the last ∼6000. The reason for this is not fully understood and further testing is required; one hypothesis is that the buffer between the FPGA and the PC had to be cleared. Hence the first recordings were actually made before the test was started (and before the jammer was turned on. The readings were placed in the communications buffer by the FPGA and not received by the PC as the Python script was not running. Once the script was started the buffer was emptied and eventually caught up with the current data. As the jammer was not turned on between



(a) Data from Fig. 7.10 with a ten-point moving average applied.

(b) Data from Fig. 7.10 with a fifty-point moving average applied.

Figure 7.11: The result of applying a ten or fifty point moving average window to the raw data.

tests the result would be noisier. It is noticeable that when the solution is more stable (after sample 4000) the number of times the algorithm resets to 161 is also reduced, indicating that the algorithm does not detect large jumps in the solution as often.

While it is not possible to measure the power consumption of just the FPGA, it was of interest to calculate the dynamic power of the program being run, once it had been placed on an actual device. First the power consumption of the entire DE1 board was measured, with no program loaded. This measurement should be the static power consumption of the board, including all peripherals, giving a baseline. Following this, the program was loaded onto the FPGA and the power consumption was measured again. The difference between the two measurements provides an estimate of the dynamic power. In this case the current was measured as 0.262 A unprogrammed, and 0.2795 A once programmed. Therefore the dynamic current was approximately 0.0175 A. As the power supply was 12 V, this translates to a dynamic power consumption of 0.21 W. This is slightly higher than the estimate given by the simulator.

The University of Bath has a system that uses the MUSIC algorithm presented in Chapter 2 to calculate the direction of arrival. The system comprises a PC with an Intel Xeon processor, plus the RF front end required to process the signal, with a total power consumption of around 300 W. This system is capable of producing a solution approximately once every 20 ms, with no averaging. Hence the system described in this thesis can produce a solution significantly faster for a fraction of the power consumption, although the quality of the solution cannot be compared without further experimentation.

7.6 Conclusions

Previous Chapters of this Thesis have discussed using a bio-inspired algorithm called Invasive Weed Optimisation to decrease the time taken when searching for jammers. It was then proposed that jammers could be tracked by modifying this algorithm. Unfortunately the performance (in terms of accuracy) of this new algorithm could not be verified in field tests as it was implemented in Python on a PC and there was a significant bottleneck in the PC-beamformer communications. To circumvent this bottleneck it was decided to implement the tracking algorithm on an embedded platform, thus increasing the communications rate.

It was decided that an FPGA would be used as research has shown that they have the best performance in terms of Watts per computation. It also offered the greatest research potential. The algorithm was implemented using HDL on an FPGA, with some modifications, and was tested in both simulation and in field tests.

In simulation it was found that the performance of the HDL implementation was slightly worse than the Python implementation. However, it was still capable of tracking both steps and ramps. The HDL was then fit to a variety of low power FPGAs and the resource and power consumption was assessed. Resource use was generally low, indicating that smaller (and therefore cheaper) devices could be used. When devices designed for their low power consumption were chosen, power consumption was good considering the clock speed, which was typically faster than 70 MHz. However, power consumption is still higher than some other embedded options, such as microprocessors.

Finally, the entire system was tested in the field. The tests were not exhaustive but it was shown to be capable of tracking a moving jammer in real time. The result was noisy, the reason for which is not yet understood. However, using a 10- or 50-point moving average significantly improved the quality of the result.

The algorithm could do with some improvements. Simulations imply that it performs worse than the Python-based implementation (although this is not certain as no Python-based systems were tested in the field). There are a number of reasons that this could be the case. The most significant changes were the loss of precision when using fixed point number representations, and the changes to the random number generation. As the total time required for one iteration was approximately 620 μs (in simulation), and the time taken for the calculations was only 250 μs, there is scope to increase the time taken by calculations without decreasing the overall speed of the program's operation. There is also scope for increasing the size of the hardware, as the fitter results showed that the devices were not fully used. However, this will increase the overall power consumption by increasing the amount of switching and therefore the dynamic power consumption.

The solution appears to need some averaging before it can produce a good answer. Using a moving average increases computation (and therefore power consumption) but does not decrease the rate at which solutions are produced. Implementing averaging may negate the problems discussed above with regard to the accuracy, but further testing would be required to verify it.

Chapter 8

Conclusions

The aim of this research was to create a proof-of-concept system for detecting, locating, and potentially mitigating GNSS interference. Two use cases were proposed. In the first, the system is mounted on the underside of a UAV and used to scan for interference sources. Information would be relayed back to a user on the ground who could then find the jammer. The second use case is a static, ground based system. It could be sited, for example, at an airport and used to detect vehicles carrying jammers. The jamming detection system would be able to report back the location of the jamming signal to another system, allowing it to mitigate the jammer.

At the beginning of this Thesis, three research questions were posed:

1. Is it possible to detect, locate, and/or mitigate GNSS jamming using a low power, low cost system?
2. Is it possible to find out information about the jammer in question?
3. What methods can be used to speed up the rate at which it locates the jamming, without significantly increasing the power consumption of the system?

At this point, it is possible to reflect upon the extent to which these questions have been answered.

1. Is it possible to detect, locate, and/or mitigate GNSS jamming using a low power, low cost system?

In Chapter 2 an array was designed with these goals in mind. The literature was reviewed to find a localisation method that would minimise power consumption and cost. It was decided to use the angle of arrival of the signal as it can be achieved without complicated computation. It also did not necessarily require downconversion of the signal, minimising the power consumption of the RF front end. Three designs were simulated and the best one chosen. The 'best' in this case was an antenna that was physically small, minimising manufacturing costs. (All designs were simulated as if they were on FR4 substrate, which is a standard process and so costs are kept low, although it does increase the overall size of the antenna for a given array layout.) A smaller antenna is also lighter weight, making it more suitable for mounting on a UAV. This design also had a small and simple beamformer, meaning the size, weight and cost of the RF front end is kept down. Aside from the physical properties, this design also had the best radiation pattern. The peak of the signal was just

0.34° wide, giving it good precision. The peak gain of the signal was also the highest, at 48.8 dB between the main lobe and the bottom of the null. This means the antenna is sensitive to incoming signals.

In Chapter 4 an RF front end suitable for use with the designed antenna was created. The front end was designed to be reconfigurable so if previously unforeseen experiments arose, they could be carried out with no additional hardware costs. The modular, reconfigurable nature of this front end did not, however, impair its performance. Also in Chapter 4 the antenna designed in Chapter 2 was combined with the RF front end. A Python program was written to run on a PC and interface with the hardware. This complete system was demonstrated to be capable of detecting and locating a jammer by performing an exhaustive sweep. Once the sweep was complete, the highest point can be read. This gives the direction in which the jammer is found.

Overall, it has been demonstrated that it is possible to detect and locate a GPS jammer (and therefore theoretically any GNSS jammer, using the same principles) using low power, low cost hardware. While mitigation has not been demonstrated, the hardware used was suitable for mitigation. It was able to report back the location of the strongest incident signal so that a system could use this to null out interference. Alternatively the same hardware could be used, with additional software, as the components chosen had sufficient dynamic range to allow legitimate GNSS signals to pass without distortion even in the presence of interference.

2. Is it possible to find out information about the jammer in question?

The system used detection and location works by simply measuring the incident power. This method gives no information about the source of the interference. Therefore in Chapter 3 a new antenna element was designed. The element was made so that horizontal and vertical polarisations could be measured independently, allowing the system to estimate the polarisation of the received signal. Simulations indicated that the coupling between the two ports was low and the polarisation purity was good, meaning the antenna would be suitable for polarisation measurements. However, when the element was fabricated it was found to be significantly different to the simulations, to the point where polarisation measurements of GPS L1 jammers were not possible. Time was spent trying to understand why there was such a discrepancy between simulation and reality. It was determined that the problem was most likely a small air gap between the substrate and the antenna. The time required to diagnose the problem had already been significant and the additional time that would be needed to design and manufacture at least one additional iteration of the antenna, and subsequent time required to test it at facilities that were not available at the University of Bath, meant it was not viable to pursue it further. The project was shelved in favour of spending more time on optimising the angle of arrival measurements. Hence the second question of this Thesis has not been answered.

3. What methods can be used to speed up the rate at which it locates the jamming, without significantly increasing the power consumption of the system?

The initial system as described in Chapter 4 used a Python program running on a PC to perform exhaustive sweeps. This was inefficient in many ways and ran completely counter to the aims of this project, needing a whole PC in order to operate and testing many points

in order to find one answer. The first inefficiency to be removed was the search method. An exhaustive sweep is guaranteed to find the best result eventually, but will test many points in areas in which the result is unlikely to be found. In Chapter 5 bio-inspired search algorithms were used to reduce the number of positions tested, with good results. An exhaustive sweep contained 256 steps but when Particle Swarm Optimisation was used, an answer within 1.4° could be found after testing (on average) just 50 positions. The Invasive Weed Optimisation algorithm had an even better accuracy, but tested a slightly higher number of positions.

The second task was to remove the PC from the system. Two system requirements (speed and cost) were not met when a PC was used. The PC consumes a significant amount of power (around 45 W for a laptop computer) and slows the operation of the system by causing a bottleneck in the PC-hardware communications. Chapter 7 describes the process of choosing a new platform on which to run the algorithm, and the process of adapting the algorithm for the new platform.

Using an FPGA instead of a PC decreased the time taken for one iteration from around three seconds to 1.5 ms. This was sufficiently fast that a moving jammer could be tracked in real time. The whole system was tested in the field and was proved to work. Power consumption figures for the FPGA-based system are not available because only a development board was available (which used additional power compared to a more polished system with no extra components) but it is safe to say that the consumption is much lower than that of a PC. Hence by using an FPGA to run an optimisation algorithm, the speed of operation and the power consumption of the system can both be greatly improved.

8.1 Limitations of this work

The work described in this Thesis has a number of inherent limitations. Some are fundamental limitations with the method chosen, while others are caused by the design decisions in the pursuit of creating a low power, low cost solution.

For example, the choice to use the total received power instead of a mathematical super-resolution method such as MUSIC means that the jammer's power must be stronger than the noise floor. This may not be the case for all interferers. It has been shown that even weak interferers will reduce the accuracy of a position and timing fix. This system is also not capable of detecting multiple jammers and will simply detect the strongest signal. In a more realistic scenario, even if only one jammer is present there will also be multipath causing interference from multiple directions. This system relies on only the strongest signal being capable of disrupting the GNSS receiver. It also relies on the jammer radiating at 1.575 GHz. Most receivers rely on the GPS L1 signal, using other constellations and other frequency signals to increase the precision of the result. Hence for a jammer to be effective it must block at least L1, so this is a reasonable assumption. However, it assumes that the interference is intentional; a re-radiating L2 antenna would not be detected by this system. This could be solved by using a more wideband antenna or one that can receive multiple frequency bands.

Even if there is only one interference signal presence and the system is able to detect it, there are still some limitations. A decision was made early on that a low power phase shifter would be used. However, this particular device is only capable of steps of 1.4°, limiting the

accuracy of the result. The solution found by the system may also not be the best option, as the optimisation algorithm used may be faster than an exhaustive sweep but is not guaranteed to find the best solution.

8.2 Further work

8.2.1 System improvements

The components used in this system were chosen based on their performance and power consumption according to the datasheet and alternatives were not tested. It may be that using different components (perhaps components that were not available when the relevant work was being carried out) would yield better results in terms of speed or error.

There are also system elements that were not designed as they were not necessary for the operation of the system, but may improve the quality of the result. For instance, attenuators would allow for balancing of inter-component variation between signal paths, making the beam pattern closer to the theoretical. This in turn would increase the accuracy of the system.

Once the component choices have been finalised, the entire beamforming system can be miniaturised so that it is all on a single board, rather than the distributed approach taken for reasons related to easy changing of the design. It could also be fitted with some form of short range wireless communications, if the system was to be mounted on a UAV, for live updating of the jammer's estimated location.

While the polarisation measurements did not come to fruition, it may still be possible to gather more information about the signal. For instance, it is known that many jammers have a frequency sweep, or 'chirp'. A chirp detection system could be added to the hardware so that another method of fingerprinting jammers could be used.

8.2.2 Mitigation of jamming

This Thesis has described work on the detection and localisation of jammers but has not attempted any mitigation. There are a number of proposed methods of mitigating jamming. For instance, Fante *et al.* use prior knowledge of the location of GPS satellites so point narrow beams at them, thus excluding other signals such as jammers [145]. However, this requires a large beamformer and array to produce all of the required beams simultaneously. Other methods rely on computation to remove jamming-related inaccuracies in the decoder, instead of excluding unwanted signals at the antenna [146]. Based on the work in this Thesis, it should be possible to exclude jamming signals at the receiver but with minimum effort and a small array. The system tested in this research is capable of reporting a direction of arrival of an interfering signal to an external system once every 1.5 ms. In this case the system was a PC which recorded the results for later analysis, but this could also be a GPS receiver. It could use the information regarding the direction of arrival to point a null at the interferer and mitigate the jamming. Alternatively, the system could be expanded so that it carried out the mitigation itself. If this was the case the high dynamic range of the low-noise amplifiers used in the beamformer would not distort the legitimate GPS signal even in the presence of

a jammer. If, however, the information regarding the jammer's location was to be passed to an external system, the dynamic range is less important and the amplifiers can be changed to a device with a lower power consumption.

8.2.3 Path planning

Mounting the system on a UAV would allow a wide area to be searched rapidly. However, the system is only able to establish a direction of arrival and not an exact point or location. Multiple UAVs could be used to provide multiple measurements. Alternatively one UAV could make multiple measurements over time which could then be correlated. Either method could be optimised to maximise the precision of the measurements. Path planning for UAVs (albeit fixed-wing type aircraft) performing localisation tasks has been researched by Dogancay *et al.* [147], who also took into account avoiding of obstructions. This would be vital in built-up areas. Limiting the number of UAVs was also researched by Semper *et al.* [148]. Both approaches took into account the limitations of the UAVs with regard to turning speed, minimum velocity and the like. A multirotor UAV is more flexible in this regard, although maximum speed will still need to be a factor.

Chapter 9

Acknowledgements

I would like to give my thanks to all of the people who kept me going: to Alex, for the good times; to Emily, for her wisdom; to Pierre, for keeping me company; to Rhiannon, for making me take a break when I needed it; and to Heather and Jahnavi, for always being there to talk to.

I would also like to thank my first supervisor, Dr. Robert Watson. He not only found the perfect project for me, but went above and beyond to support my development throughout my PhD. Ultimately he was a good supervisor, but also a good friend.

My second supervisor, Prof. Cathryn Mitchell, receives my thanks for pushing me to be more confident.

Finally, I would like to thank Dr. Brian Nicholson for not only funding, but also being interested in, my work.

Bibliography

[1] W. Blanchard, "The genesis of the Decca navigation system," *Journal of Navigation*, vol. 68, no. 2, March 2015.

[2] W. P. Campbell, "Gee and Loran radar navigational systems World War II."

[3] "Economic impact to the UK of a disruption to GNSS," London Economics, Tech. Rep., April 2017.

[4] N. Vaughan, "GPS tracking jammers and blockers - are they legal?" accessed: 07/10/2019. [Online]. Available: https://www.vehicletrackingexperts.co.uk/gps-jammers/

[5] "Timing and synchronization for LTE-TDD and LTE-Advanced mobile networks," Symmetricom, Tech. Rep., 2013.

[6] "Major power failure affects homes and transport," *BBC*, August 2019. [Online]. Available: https://www.bbc.co.uk/news/uk-49300025

[7] "Range rover gang stole luxury cars worth £680000 by jamming owners' keyfobs as they tried to lock their cars while walking away," July 2016. [Online]. Available: https://www.dailymail.co.uk/news/article-3699304

[8] K. Collier and A. Harris, "Taxi cheats using mobile GPS jammers to steal fares," 2013. [Online]. Available: http://www.heraldsun.com.au/taxi-cheats-using-mobile-gps-jammers-to-steal-fares/news-story/567cfb3c4ba76f5b8c79d835ae9d8087

[9] D. Technologies, "How to bypass a house arrest ankle bracelet monitor," 2012. [Online]. Available: http://www.digitaltechnologies-2000.com/how-to-bypass-house-arrest-ankle-bracelet/

[10] R. H. Mitch, R. C. Dougherty, M. L. Psiaki, S. P. Powell, B. W. O'Hanlon, J. A. Bhatti, and T. E. Humphreys, "Signal characteristics of civil GPS jammers," *Proceedings of ION GNSS 2011*, pp. 20–23, 2011.

[11] "Detecting and Geolocating the Source of GPS Interference," Feb. 2017, Chronos Technology.

[12] "Detecting Rogue GPS Antennas," Feb. 2017, Chronos Technology.

[13] "No jam tomorrow," Mar. 10 2011. [Online]. Available: https://www.economist.com/node/18304246

[14] C. Matyszczyk, "Truck driver has GPS jammer, accidentally jams Newark airport," August 2013, accessed 07/08/19. [Online]. Available: https://www.cnet.com/news/truck-driver-has-gps-jammer-accidentally-jams-newark-airport/

[15] C. Technology, "SENTINEL project report on GNSS vulnerabilities," 2014.

[16] D. Borio, F. Dovis, H. Kuusniemi, and L. Lo Presti, "Impact and detection of GNSS jammers on consumer grade satellite navigation receivers," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1233–1245, 2016.

[17] D. Borio, C. O'Driscoll, and J. Fortuny, "Jammer impact on Galileo and GPS receivers," in *2013 International Conference on Localization and GNSS*, 2013, Conference Proceedings, p. 6.

[18] "Economic impact to the UK of a disruption to GNSS," 2017, London Economics and Innovate UK.

[19] A. C. O'Connor, M. P. Gallaher, K. Clark-Sutton, D. Lapidus *et al.*, "Economic benefits of the Global Positioning System (GPS)," 2019, RTI International, sponsored by the National Institute of Standards and Technology.

[20] Symmetricom, "Power Utilities: Mitigating GPS Vulnerabilities and Protecting Power Utility Network Timing," 2013.

[21] "Taxi driver convicted," 2014. [Online]. Available: https://www.acma.gov.au/theACMA/taxi-driver-convicted

[22] B. Marshall, "Analysing GPS jamming incidents at the UK border," 2017, available: https://www.gps.gov/governance/advisory/meetings/2017-06/marshall.pdf.

[23] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, June 2016.

[24] "UT Austin researchers successfully spoof an $80 million yacht at sea," July 2013, available: https://news.utexas.edu/2013/07/29/ut-austin-researchers-successfully-spoof-an-80-million-yacht-at-sea/.

[25] D. Goward, "GPS jamming and spoofing reported at Port of Shanghai," August 2019, available: https://www.maritime-executive.com/editorials/gps-jamming-and-spoofing-at-port-of-shanghai.

[26] M. Fairbanks and B. Cockshott, "Maritime resilience and integrity of navigation," August 2019.

[27] "Above us only stars: Exposing GPS spoofing in Russia and Syria," C4ADS, Tech. Rep., 2019.

[28] D. Schmidt, "A survey and analysis of the GNSS spoofing threat and countermeasures," *ACM Computing Surveys*, vol. 48, no. 4, p. 64:1, 2016.

[29] C. Bonebrake and L. R. O'Neil, "Attacks on GPS time reliability," *IEEE Security & Privacy*, vol. 12, no. 3, pp. 82–84, 2014.

[30] U. Hunkeler, J. Colli-Vignarelli, and C. Dehollain, "Effectiveness of GPS-jamming and counter-measures," in *International Conference on Localization and GNSS, 2012*, Conference Proceedings, pp. 1–4.

[31] A. Tani and R. Fantacci, "Performance evaluation of a precorrelation interference detection algorithm for the GNSS based on nonparametrical spectral estimation," *IEEE Systems Journal*, vol. 2, no. 1, pp. 20–26, 2008.

[32] A. Garofalo, C. D. Sarno, L. Coppolino, and S. D'Antonio, "A GPS spoofing resilient WAMS for smart grid," *Lecture Notes in Computer Science*, vol. 7869, pp. 134–147, 2013.

[33] Microsemi, "Microsemi 5071a primary frequency standard datasheet," 2014.

[34] R. L. Haupt, "Phase-only adaptive nulling with a genetic algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 6, pp. 1009–1015, 1997.

[35] E. Lloyd, "Report on Sennybridge GPS jamming trials, 16th-17th August 2017," Chronos Technology, Tech. Rep., 2017, available on request from Chronos Technology.

[36] S. Liu, D. Li, B. Li, and F. Wang, "A compact high-precision GNSS antenna with a miniaturized choke ring," *IEEE Antennas and Wireless Propagation Letters*, vol. 16, 2017.

[37] "CTL3510 GPS Jammer Detector," 2017. [Online]. Available: http://www.gps-world.biz/products/gnss-interference-detection/products-solutions/ctl-3510

[38] "CTL3520 Handheld GPS Jammer Detector and Locator," 2017. [Online]. Available: http://www.gps-world.biz/products/gnss-interference-detection/products-solutions/ctl-3520

[39] R. Bauernfeind, T. Kraus, A. Sicramaz Ayaz, D. Dötterböck, and B. Eissfeller, "Analysis, detection and mitigation of incar GNSS jammer interference in intelligent transport systems," September 2012.

[40] G. Kar, M. Gruteser, Y. Wang, Y. Chen, H. Mustafa, W. Xu, and T. Vu, "Detection of on-road vehicles emanating GPS interference," in *ACM Conference on Computer and Communications Security*, Conference Proceedings, pp. 621–632.

[41] "JammerCam," accessed 07/08/19. [Online]. Available: https://www.gps-world.biz/products/gnss-interference-detection/jammercam

[42] J. Lindström, D. Akos, O. Isoz, and M. Junered, "GNSS interference detection and localization using a network of low cost front-end modules," *Proceedings of the 20th International Technical Meeting of the Satellite Division of the Institute of Navigation*, pp. 1165–1172, 2007.

[43] E. M. Lloyd and R. J. Watson, "An array antenna for low power localisation of GPS interference," in *European Conference on Antennas and Propagation (EuCAP) 2018*, 2018, Conference Proceedings.

[44] ——, "Development of a reconfigurable modular GPS beamformer for design and test," *2017 Loughborough Antennas and Propagation Conference, LAPC 2017*, 2017.

[45] ——, "Using a bio-inspired algorithm for efficient angle-of-arrival estimation of GNSS jammers," in *European Conference on Antennas and Propagation (EuCAP) 2019*, 2019, Conference Proceedings.

[46] ——, "Comparison of adaptive null-steering algorithms for low power GNSS phased arrays," in *European Conference on Antennas and Propagation (EuCAP) 2020*, 2020.

[47] N. Okello, "Emitter geolocation with multiple UAVs," in *9th International Conference on Information Fusion*, Conference Proceedings, pp. 1–8.

[48] N. Okello and D. Musicki, "Emitter geolocation with two UAVs," in *Information, Decision and Control*. IEEE, Conference Proceedings.

[49] ——, "Measurement association for emitter geolocation with two UAVs," in *Information Fusion*. IEEE, Conference Proceedings.

[50] N. Okello, F. Fletcher, D. Musichi, and B. Ristic, "Comparison of recursive algorithms for emitter localisation using TDOA measurements from a pair of UAVs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 3, p. 9, 2011.

[51] J. A. Bhatti, T. E. Humphreys, and B. M. Ledvina, "Development and demonstration of a TDOA-based GNSS interference signal localization system," in *Proceedings of the 2012 IEEE/ION position, location and navigation symposium*, July, Conference Proceedings.

[52] H. Hmam, "Scan-based emitter passive localization," *IEEE Transactions on Aerospace & Electronic Systems*, vol. 43, no. 1, pp. 36–55, 2007.

[53] Q. L. Jing Liang, "RF emitter location using a network of small unmanned aerial vehicles (SUAVs)," in *2011 IEEE International Conference on Communications (ICC)*, Conference Proceedings.

[54] T. Collins, "gr-doa: GNU Radio Direction Finding," Online, 2017, conference presentation at GRCon17.

[55] B. Lu, B. Wen, Y. Tian, and R. Wang, "Analysis and calibration of crossed-loop antenna for vessel DOA estimation in HF radar," *IEEE Antennas and Wireless Propagation Letters*, vol. 17, no. 1, November 2017.

[56] R. Schmidt, "Multiple emitter location and signal parameter estimation," *Antennas and Propagation, IEEE Transactions on*, vol. 34, no. 3, pp. 276–280, 1986.

[57] R. Adve, "Direction of arrival estimation," 2013, accessed: 21/12/19.

[58] M. P. Alex B. Gershman, Michael Ruebsamen, "One- and two-dimensional direction-of-arrival estimation: An overview of search-free techniques," *Signal Processing*, vol. 90, no. 5, pp. 1338–1349, 2010.

[59] A. Cichocki, "Unsupervised learning algorithms and latent variable models: PCA/SVD, CCA/PLS, ICA, NMF, etc," in *Academic Press Library in Signal Processing*. Academic Press, Sep 2013, ch. 21.

[60] B. Friedlander, "The root-MUSIC algorithm for direction finding with interpolated arrays," *Signal Processing*, vol. 30, no. 1, 1993.

[61] F. Belloni, A. Richter, and V. Koivunen, "Extension of root-MUSIC to non-ULA array configurations," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, 2006, Conference Proceedings, pp. IV897–IV900.

[62] M. G. Amin, X. Wang, Y. D. Zhang, F. Ahmad, and E. Aboutanios, "Sparse arrays and sampling for interference mitigation and DOA estimation in GNSS," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1302–1317, 2016.

[63] R. Roy and T. Kailath, "ESPRIT - Estimation of Signal Parameters via Rotational Invariance Techniques," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 7, 1989.

[64] A. Badawy, T. Khattab, D. Trinchero, T. Elfouly, and A. Mohamed, "A simple cross correlation switched beam system (XSBS) for angle of arrival estimation," *IEEE Access*, vol. 5, pp. 3340 – 3352, 2017.

[65] V. F. Fusco and H. I. Cantù, "Self-aligning wireless link using modulated backscatter," *IET Microwaves, Antennas and Propagation*, vol. 4, no. 9, 2010.

[66] Y. Fan and Y. Rahmat-Samii, "Microstrip antennas integrated with electromagnetic band-gap (EBG) structures: a low mutual coupling design for array applications," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 10, pp. 2936–2946, 2003.

[67] M. I. Skolnik, *Radar Handbook*, 1st ed. McGraw-Hill, 1970, pp. 9.21.

[68] R. C. Hansen and R. E. Collin, *Small Antenna Handbook*, 11th ed. Wiley, 2011.

[69] B. Sheleg, "A matrix-fed circular array for continuous scanning," *Proceedings of the IEEE*, vol. 56, no. 11, pp. 2016–2027, 1968.

[70] G. Byun, H. Choo, and S. Kim, "Improvement of pattern null depth and width using a curved array with two subarrays for CRPA systems," *Antennas and Propagation, IEEE Transactions on*, vol. 63, no. 6, pp. 2824–2827, 2015.

[71] "Monopulse antennas," accessed: 21/03/20. [Online]. Available: https://www.microwaves101.com/encyclopedias/monopulse-antennas

[72] G. Jones, "Mobile menace: why SDR poses such a threat," *Network Security*, vol. 2012, no. 6, p. 3, 2012.

[73] A. Jafarnia-Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "GPS vulnerability to spoofing threats and a review of antispoofing techniques," *International Journal of Navigation and Observation*, 2012.

[74] D. J. Jwo, F. C. Chung, and K. L. Yu, "GPS/INS integration accuracy enhancement using the interacting multiple model nonlinear filters," *Journal of Applied Research and Technology*, vol. 11, no. 4, 2013.

[75] A. Cavaleri, B. Motella, M. Pini, and M. Fantino, "Detection of spoofed GPS signals at code and carrier tracking level," in *Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, 2010, Conference Proceedings.

[76] J. Magiera and R. Katulski, "Detection and mitigation of GPS spoofing based on antenna array processing," *Journal of Applied Research and Technology*, vol. 13, no. 1, pp. 45–57, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1665642315300043

[77] M. Born and E. Wolf, *Principles of Optics*, 2nd ed. Pergamon Press, 1964, book section 1. Basic properties of the electromagnetic field, p. 808.

[78] R. Finkele, A. Schreck, and G. Wanielik, "Polarimetric road condition classification and data visualisation," in *International Geoscience and Remote Sensing Symposium, IGARSS*, vol. 3, 1995, pp. 1786–1788.

[79] A. L. Durkee, D. Metcalfe, and W. H. Tidd, "White alice system - design and performance," *IRE Transactions on Communications Systems*, vol. 7, no. 4, 1959.

[80] C. A. Balanis, *Antenna theory: analysis and design*, 3rd ed. Wiley-Blackwell, 2005, ch. Microstrip Antennas.

[81] M. Abbaspour and H. R. Hassani, "Wideband planar patch antenna array on cylindrical surface," *Antennas and Wireless Propagation Letters, IEEE*, vol. 8, pp. 394–397, 2009.

[82] M. Y. W. Chia and Z. N. Chen, *Broadband planar antennas: design and applications*. Wiley, 2006, ch. Broadband Microstrip Patch Antennas.

[83] D. M. Pozar, "Microstrip antenna aperture-coupled to a microstripline," *Electronics Letters*, vol. 21, no. 2, pp. 49–50, January 1985.

[84] T. A. Denidni and M. Hotton, "Design of aperture-coupled microstrip antenna for applications in wireless communications," in *SMBO/IEEE MTT-S International Microwave and Optoelectronics Conference*, 1999.

[85] "Premix group," 2017. [Online]. Available: http://www.premixgroup.com/

[86] L. Josefsson and P. Persson, *Conformal array antenna theory and design*, ser. The IEEE Press Series on Electromagnetic Wave Theory. IEEE Press, 2006.

[87] R. Huang, H. Liao, and G. Zhang, "SOI CMOS technology for RF/MMIC applications - yes or no?" in *Proceedings of 35th European Solid-State Device Research Conference*, 2005.

[88] Rogers RO-4000 Series laminates. Accessed 18/01/20. [Online]. Available: https://rogerscorp.com/en/advanced-connectivity-solutions/ro4000-series-laminates

[89] T. Vu, "Method of null steering without using phase shifters," *IEE Proceedings: Microwaves, Optics and Antennas*, vol. 131, no. 4, 1984.

[90] "AD8341 1.5 GHz to 2.4 GHz RF Vector Modulator Data Sheet," p. 21, 2017, Analog Devices.

[91] "MCP1755 300 mA 16V High Performance LDO," 2012.

[92] "PE44820 UltraCMOS RF Digital Phase Shifter 1.7-2.2 GHz," 2016.

[93] "Skyworks SKY12349-362LF 0.7 - 4GHz five-bit digital attenuator," 2011.

[94] "Peregrine Semiconductor PE43712 UltraCMOS RF Digital Step Attenuator, 9kHz - 6 GHz," 2018.

[95] "Precision chip attenuators PAT series," https://www.susumu.co.jp/usa/product/category.php?cid=16, accessed: 26/02/2020.

[96] D. M. Pozar, *Microwave engineering*, 3rd ed. John Wiley and Sons, 2005.

[97] "Ultra-Small Ceramic Power Splitter/Combiner SCN-2-19+, 2 way-0°," MiniCircuits.

[98] "PE42442 UltraCMOS SP4T RF Switch 30 MHz - 6 GHz," 2017.

[99] "HMC345ALP3E, GaAs MMIC SP4T non-reflective positive control switch, DC-8 GHz."

[100] "Microwaves 101 Encyclopedia: Mitred bends." [Online]. Available: https://www.microwaves101.com/encyclopedias/mitered-bends

[101] "Ultra-Small Ceramic Power Splitter/Combiner QCN-2-19+, 2 way-90°," MiniCircuits.

[102] "MGA-62563 Current-Adjustable, Low Noise Amplifier Data Sheet," p. 19, 2011.

[103] "50MHz to 3GHz RF power detector with 60dB dynamic range," 2004.

[104] OM13056 LPCXpresso board for LPC1549. https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc1500-cortex-m3/lpcxpresso-board-for-lpc1549:OM13056. Accessed: 27/02/2020.

[105] "Wireless telegraphy act 2006," online, accessed 20/10/2020. [Online]. Available: https://www.legislation.gov.uk/ukpga/2006/36/contents

[106] D. T. Pham and D. Karaboga, *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks.* London: Springer, 2000.

[107] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas & Propagation*, vol. 52, no. 3, pp. 771–780, 2004.

[108] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A comparison of particle swarm optimisation and the genetic algorithm," in *46th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, April 2005.

[109] S. Volz, *Microscale and nanoscale heat transfer*. Springer, 2006, ch. Monte Carlo Method.

[110] M. Crepinsek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: a survey," *ACM Computing Surveys*, 2013.

[111] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Nov 1995, Conference Proceedings.

[112] A. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, no. 4, pp. 355–366, 2006.

[113] M. M. Roshanaei, "Adaptive beamforming using a novel numberical optimisation algorithm," *IET Microwaves, Antennas & Propagation*, vol. 3, no. 5, pp. 765–774, 2009.

[114] "Statistical methods for convergence detection of multi-objective evolutionary algorithms," *Evolutionary Computation*, vol. 17, no. 4, 2009.

[115] D. I. Olcan, R. M. Golubovic, and B. M. Kolundzija, "On the efficiency of particle swarm optimizer when applied to antenna optimization," in *2006 IEEE Antennas and Propagation Society International Symposium*, Conference Proceedings, pp. 3297–3300.

[116] T. Peters, "listsort.txt," accessed 11/11/2019. [Online]. Available: https://github.com/python/cpython/blob/master/Objects/listsort.txt

[117] C. Arthur, "Thousands using GPS jammers on UK roads pose risks, say experts," February 2013, accessed 07/08/19. [Online]. Available: https://www.theguardian.com/technology/2013/feb/13/gps-jammers-uk-roads-risks

[118] A. Lindgren and K. F. Gong, "Position and velocity estimation via bearing observations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 14, July 1978.

[119] Z. Yiyu and S. Zhongkang, "Passive location and tracking of 3-D moving emitter using DOA and TOA measurements."

[120] C. M. Rose, "Augmented passive tracking of moving emitter," U.S. Patent 7 626 538, Dec 01, 2009.

[121] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation*, May 1998, Conference Proceedings.

[122] Y.-Y. Bai, "A hybrid IWO/PSO algorithm for pattern synthesis of conformal phased arrays," *IEEE Transactions on Antennas & Propagation*, vol. 61, no. 4, pp. 2328–2333, 2013.

[123] M. Thida, H.-L. Eng, D. N. Monekosso, and P. Remagnino, "A particle swarm optimisation algorithm with interactive swarms for tracking multiple targets," *Applied Soft Computing*, vol. 13, June 2013.

[124] M. Thida, P. Remagnino, and H.-L. Eng, "A particle swarm optimization approach for multi-objects tracking in crowded scene," in *IEEE 12th International Conference on Computer Vision Workshops*, 2009.

[125] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu, "Sequential particle swarm optimization for visual tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[126] J. R. Leigh, *Control theory.* IEEE, 2004.

[127] S. Salous, *Radio propagation measurement and channel modelling.* Hoboken: John Wiley and Sons Inc., 2013.

[128] (2018) LPC81xM 32-bit ARM Cortex-M0+ microcontroller. Datasheet; Accessed 27/12/19. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/LPC81XM.pdf

[129] A. Beasley, "Exploring the benefits and implications of dynamic partial reconfiguration using field programmable gate array - system on chip architectures," ch. 5, pp. 108–128, Ph.D. dissertation.

[130] "Cortex-M3," ARM, accessed: 16/07/2019. [Online]. Available: https://developer.arm.com/ip-products/processors/cortex-m/cortex-m3

[131] S. M. Noor and E. John, "Performance and energy evaluation of ARM Cortex variants for smart cardiac pacemaker application," in *The 2nd International Conference on Biomedical Engineering and Sciences (BIOENG'16)*, 2016, Conference proceedings.

[132] "ARM11 vs Cortex A8 vs Cortex A9 - Netbooks processors." [Online]. Available: https://web.archive.org/web/20110719103301/http://www.eeejournal.com/2009/12/arm11-vs-cortex-a8-vs-cortex-a9.html

[133] G. Coley and J. Kridner, *BeagleBone Black System Reference Manual*, accessed: 17/07/2019. [Online]. Available: https://github.com/beagleboard/beaglebone-black/wiki/System-Reference-Manual

[134] B. Benchoff, "Benchmarking the Raspberry Pi 2," accessed: 16/07/2019. [Online]. Available: https://hackaday.com/2015/02/05/benchmarking-the-raspberry-pi-2/

[135] *AM437x Sitara Processors*, 2019.

[136] J. Kridner and J. Coley, *System Reference Manual*, Apr 2019, section 6.1.7.

[137] J. Geerling, "Power consumption benchmarks," accessed: 31/12/2019. [Online]. Available: https://www.pidramble.com/wiki/benchmarks/power-consumption

[138] M. J. Hibbard, E. R. Peskin, and F. Sahin, "FPGA implementation of particle swarm optimization for Bayesian network learning," *Computers and Electrical Engineering*, vol. 39, 2013.

[139] D. M. Muñoz, C. H. Llanos, L. dos Santos Coelho, and Ayala-Rincón, "Accelerating the Shuffled Frog Leaping algorithm by parallel implementations in FPGAs," in *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, Sept 2010.

[140] J. S. Malik and A. Hemani, "Gaussian random number generation: a survey on hardware architectures," *ACM Computing Surveys*, vol. 49, December 2016.

[141] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM Transactions on Modeling and Computer Simulations*, 1998.

[142] W. Yuan and Z. Xu, "FPGA based implementation of low-latency floating-point exponential function," in *IET International Conference on Smart and Sustainable City 2013*, 2013.

[143] Modelsim. https://www.mentor.com/products/fv/modelsim/. Accessed: 06/02/2020.

[144] "Cyclone V FPGAs," accessed: 27/10/2020. [Online]. Available: https://www.intel.co.uk/content/www/uk/en/products/programmable/fpga/cyclone-v.html

[145] R. L. Fante and J. J. Vaccaro, "Wideband cancellation of interference in a GPS receive array," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 2, pp. 549–564, 2000.

[146] A. Giremus, J. Y. Tourneret, and A. Doucet, "A particle filter to mitigate jamming for GPS navigation," in *IEEE/SP 13th Workshop on Statistical Signal Processing*, 2005, pp. 1298–1303.

[147] K. Dogancay, "UAV path planning for passive emitter localization," *IEEE Transactions on Aerospace & Electronic Systems*, vol. 48, no. 2, pp. 1150–1150, 2012.

[148] J. L. C. Sean R. Semper, "Decentralized geolocation and optimal path planning using limited UAVs," in *12th International Conference on Information Fusion*, Conference Proceedings.

Appendices

Appendix A

Particle Swarm Optimisation Python code

```
1  import random
2  import csv
3  import sys
4  import copy
5  from numpy import linspace, transpose
6
7  Population = 10
8  MaxIterations = 100
9  CostFunLen = 256
10
11 SEARCHMAX = 255
12 SEARCHMIN = 0
13
14 DELTAT = 1
15 Threshold = 0.05
16 W = 1.4 #Inertia factor
17 C1 = 2.0 #Self confidence factor
18 C2 = 2.5 #Swarm confidence factor
19 ResultsFilename = 'C:/Results/Beescript/BeesVsWeeds/dump.csv'
20 CallNumber = "1"
21
22 class Particle:
23    Number = None
24    def __init__(self, Number):
25       self.Number = Number
26    Cost = 1000
27    Rank = None
28    Phase1 = None
29    Velocity = None
```

```
30      BeeBestCost = 1000
31      BeeBestPos = None
32
33  inputs = sys.argv
34  if len(inputs) > 1:
35      ResultsFilename = inputs[1]
36      CallNumber = inputs[2]
37      W = float(inputs[3])
38      C1 = float(inputs[4])
39      C2 = float(inputs[5])
40
41  #Open the objective function
42  filepath = 'H:/dos/Python/Optimisation/Notuseful/RealCostFun.csv'
43
44  yt = []
45  with open(filepath, 'rb') as f:
46      reader = csv.reader(f, delimiter = ",")
47      yt = list(reader)
48
49  for i in range(0, CostFunLen):
50      yt[0][i] = float(yt[0][i])
51  CostFunction = yt[0]
52
53  transpose(CostFunction)
54  x = linspace(0, len(CostFunction), len(CostFunction))
55
56  #Generate the initial population
57  Bees = []
58  for i in range(0, Population):
59      NewBee = Particle(i)
60      NewBee.Phase1 = random.randint(0, CostFunLen-1)
61      temp = CostFunLen / 2
62      NewBee.Velocity = random.randint(-temp, temp)
63      NewBee.Cost = CostFunction[int(NewBee.Phase1)]
64      Bees.append(NewBee)
65
66  #Initialise variables used in the loop
67  SolutionFound = False
68  SwarmBestCost = 1000
69  SwarmBestPosition = None
70  iterations = 0
71  Bests = [10000,10000,10000]
72
```

```
73  while (iterations < MaxIterations):
74     #Step 1: Generate positions/determine cost
75     OldBees = copy.copy(Bees)
76     OldBees.sort(key = lambda x: x.Cost)
77
78     Bests[0] = Bests[1] #3 element shift register
79     Bests[1] = Bests[2]
80     Bests[2] = OldBees[0].Cost
81     Average = (Bests[0] + Bests[1] + Bests[2])/3
82
83     #Check to see if there's a new swarm best
84     if OldBees[0].Cost < SwarmBestCost:
85        SwarmBestCost = OldBees[0].Cost
86        SwarmBestPosition = OldBees[0].Phase1
87
88     if Average < 0:
89        if SwarmBestCost < 0:
90           if SwarmBestCost > Average * (1 + Threshold): #If they're
                both negative, the test needs to be upside down.
91              SolutionFound = True
92
93              break
94           else:
95              SolutionFound = False
96        else: #If average is negative but swarmbestcost isn't, ...
              This is impossible.
97           SolutionFound = False
98     else: #If the average is positive
99
100       if SwarmBestCost > Average * (1 - Threshold):
101          SolutionFound = True
102          break
103       else:
104          SolutionFound = False
105
106    #Step 2: Update velocity
107    for i in range(Population):
108       #Check to see if you've got a new local best
109       if Bees[i].Cost < Bees[i].BeeBestCost:
110          Bees[i].BeeBestPos = Bees[i].Phase1
111          Bees[i].BeeBestCost = Bees[i].Cost
112
113       #Generate the new velocity
```

```
114         #Current motion
115         Term1 = W * Bees[i].Velocity
116 
117         #Particle memory influence
118         bob = (Bees[i].BeeBestPos - Bees[i].Phase1)/DELTAT
119         if bob <= 0:
120            temp1 = bob
121            temp2 = 1
122         else:
123            temp1 = 0
124            temp2 = bob
125         Term2 = C1 * random.randint(temp1,temp2)
126 
127         #Swarm influence
128         bob = (SwarmBestPosition - Bees[i].Phase1)/DELTAT
129         if bob <= 0:
130            Term3 = C2 * (bob - (random.random() * bob)) #need to make
                random.random slightly higher than bob
131         else:
132            Term3 = (random.random() * bob)
133 
134         Bees[i].Velocity = Term1 + Term2 + Term3
135         if Bees[i].Velocity > 256:
136            Bees[i].Velocity = 256
137         if Bees[i].Velocity < -256:
138            Bees[i].Velocity = -256
139 
140         #Step 3: Update position...
141         Bees[i].Phase1 = Bees[i].Phase1 + Bees[i].Velocity*DELTAT
142         Bees[i].Phase1 = int(round(Bees[i].Phase1))
143 
144         while Bees[i].Phase1 < SEARCHMIN:
145            Bees[i].Phase1 = Bees[i].Phase1 + (SEARCHMAX - SEARCHMIN)
146            Bees[i].Phase1 = Bees[i].Phase1 + SEARCHMIN
147         while Bees[i].Phase1 > SEARCHMAX:
148            Bees[i].Phase1 = Bees[i].Phase1 - (SEARCHMAX - SEARCHMIN)
149            Bees[i].Phase1 = Bees[i].Phase1 + SEARCHMIN
150 
151         #...(and therefore the cost)
152         Bees[i].Cost = CostFunction[int(Bees[i].Phase1)]
153      iterations = iterations + 1
154 
155 #Save results to a file
```

```
156 TotalBees = (iterations + 1) * 10
157 F_Input = open(ResultsFilename, 'a')
158 line = [str(SwarmBestPosition), ',', str(TotalBees), ',', str('W'),
        ',', str('C1'), ',', str(C2), ',', CallNumber, '\n']
159 F_Input.writelines(line)
160 F_Input.close()
```

Appendix B

Invasive Weed Optimisation Python code

```
#Any libraries I need to import
import random
import csv
import sys
import subprocess
import string

from datetime import datetime

startTime = datetime.now()

SWITCH = "SWITCH"
SHIFTER = "SHIFTER"
ATTENUATOR = "ATTENUATOR"
READ = "READ"

InitialPopulation = 10
MaxIterations = 40
MaxPopulation = 20
MaxSeeds = 5
MinSeeds = 0
ModIndex = 3 #Set to 3
SigmaInitial = 50 #Initial standard deviation
SigmaFinal = 1 #Final standard deviation
InitialSearchAreaMin = 0
InitialSearchAreaMax = 360
Range = 128
TotalWeeds = 0
TargetCost = 2500
```

```
30 TargetTolerance = 5 #Allows a 5% increase between generations
31 BestCost = None
32 WorstCost = None
33 dy = MaxSeeds-MinSeeds
34
35 #Define a weed object
36 class Weed:
37    Number = None
38    Cost = 10000
39    Rank = None
40    NoOfSeeds = None
41    Phase1 = None
42    Phase2 = None
43    def __init__(self, Number):
44       self.Number = Number
45
46 #Compose commands for sending to the ARM chip
47 def ComposeCommand(Type, Address, Value):
48    Address = str(Address) #Zero pad the address
49    if len(Address) == 1:
50       Address = '0' + Address
51    #Get the right command letter
52    if Type == SWITCH:
53       Type = "W"
54    elif Type == SHIFTER:
55       Type = "H"
56       Value = int(LookupTable[Value][1]) #Check for OPT bit if it's
             a shifter
57    elif Type == ATTENUATOR:
58       Type = "A"
59
60    Value = str(Value) #Convert value to string so I can manipulate
          it
61    if len(Value) == 1:
62       Value = "00" + Value #Zero pad the value to make it a
             consistent length
63    elif len(Value) == 2:
64       Value = '0' + Value
65
66    Output = Type + "␣" + Address + "␣" + Value
67
68    return Output
69
```

```
70  def ReadADCs():
71      Command = "R␣00␣000"
72      ser.write(Command)
73      reply = ""
74      ella = ser.read(1)
75
76      while (ella != "B"):
77          reply = reply + ella
78          ella = ser.read(1)
79      maxime = string.split(reply)
80      sigma = maxime[0]
81      delta = maxime[1]
82
83      return sigma, delta
84
85  #Sends 2 phases to the arm chip and then does a read request.
86  def CalculateCost(Frank):
87      Command = ComposeCommand("H", 1, Frank.Phase2)
88      ser.write(Command)
89      #Now send a read request
90      Sigma, Delta = ReadADCs()
91
92      cost = int(Sigma) - int(Delta)
93      return cost, Sigma, Delta
94
95
96  def CalculateRange(Iteration):
97      #This calculates the range over which new seeds can sprout.
            Depends on the iteration and
98      # SigmaInitial/SigmaFinal.
99      NewRange = float((MaxIterations - Iteration)**ModIndex)/float(
            MaxIterations**ModIndex)
100     NewRange = NewRange * (SigmaInitial - SigmaFinal)
101     NewRange = NewRange + SigmaFinal
102     NewRange = int(NewRange)
103     return NewRange
104
105 def GenerateWeed(Centres):
106     NewWeed = Weed(1)
107     Lows = [x-Range for x in Centres] #Lows is a 2 element array of
            the lowest number a phase can be
108     Highs = [x+Range for x in Centres] #Highs is the same
109     NewWeed.Phase1 = 0
```

```
110     NewWeed.Phase2 = Wrapto255(random.randint(Lows[1], Highs[1]))
111 
112     return NewWeed
113 
114 def Wrapto255(Angle):
115     while Angle > 255:
116         Angle = Angle - 255
117 
118     while Angle < 0:
119         Angle = Angle + 255
120 
121     return Angle
122 
123 try:
124     import serial
125 except:
126     print "Failed to import serial\n"
127     raw_input("Press enter to exit")
128     exit()
129 
130 print "Libraries imported"
131 
132 #Connect to the ARM chip
133 ser = serial.Serial("COM5", timeout = 1, baudrate = 115200)
134 print ser.name
135 
136 F_LUT = open("H:/dos/Python/Optimisation/PE44820LUT.csv", "r") #
    Open the lookup table
137 debbie = F_LUT.readlines() #Debbie is the entire file
138 LookupTable = []
139 for i in range(len(debbie)):
140     LookupTable.append(string.split(debbie[i], ","))
141 
142 F_LUT.close() #Close the lookup table after I've read it.
143 
144 Weeds = []
145 print "Generating weeds"
146 #Initialise population
147 InitCentre = [128,128]
148 for i in range(0,InitialPopulation):
149     NewWeed = GenerateWeed(InitCentre)
150     Weeds.append(NewWeed)
151     TotalWeeds += 1
```

```
152
153 NoOfWeeds = InitialPopulation
154
155 Command = "H␣00␣382"
156 ser.write(Command)
157 print "Shifter␣1␣set␣to␣180"
158
159 F_input = open("H:/dos/Python/Weed_Results/Result1Dweed1fix.csv", "
      w")#Results file opened
160 line = ["Phase␣1,Phase␣2,Sigma,Delta\n"]
161 F_input.writelines(line)
162
163 #For the initial bunch of weeds, find their cost and store it
164 for i in range(NoOfWeeds):
165    Weeds[i].Cost, Sigma, Delta = CalculateCost(Weeds[i])
166    line = [str(Weeds[i].Phase1),",",str(Weeds[i].Phase2),",",Sigma,
         ",",Delta,"\n"]
167    F_input.writelines(line)
168
169 #Variables for the while loop
170 SolutionFound = False
171 Iteration = 1
172 Bests = [0,0,0] #Shift register for finding if I've found the top
      yet
173
174 while Iteration < MaxIterations:
175    Weeds.sort(key = lambda x: x.Cost, reverse = True) #Sort in
         descending order so find max sigma-delta
176    Range = CalculateRange(Iteration)
177    NoOfWeeds = len(Weeds) #Find how many weeds there are
178
179    #Cull them if there's too many weeds (just lose the worst)
180    if NoOfWeeds > MaxPopulation:
181       Weeds = Weeds[0:MaxPopulation]
182
183    NoOfWeeds = len(Weeds)
184    #3 element shift register
185    Bests[0] = Bests[1]
186    Bests[1] = Bests[2]
187    Bests[2] = Weeds[0].Cost
188
189    Average = (Bests[0] + Bests[1] + Bests[2])/3
190    temp = Average * 1.05
```

```
191     print "Best:", Bests[2], 'Target:', temp
192 
193     if (Bests[2] > (Average * 1.05)): #If it's less than the
            tolerance better, declare done
194         SolutionFound = False
195     else:
196         SolutionFound = True
197         break
198 
199     #Generate the new seeds
200     #Got to make something in the form of y = mx + c
201     #Best weed is weeds[0]
202     BestFitness = Weeds[0].Cost
203     #Worst weed is weeds[NoOfWeeds-1]
204     WorstFitness = Weeds[NoOfWeeds-1].Cost
205     #m = dy/dx
206     dx = float(BestFitness - WorstFitness)
207     print dx
208     m = dy/dx
209     c = MaxSeeds - (m*BestFitness)
210     for i in range(NoOfWeeds):
211         Weeds[i].NoOfSeeds = int((m*Weeds[i].Cost) + c) #int() does
                floor but also casts it.
212 
213     for i in range(NoOfWeeds):
214         Centre = [0, Weeds[i].Phase2]
215         for j in range(Weeds[i].NoOfSeeds):
216             NewWeed = GenerateWeed(Centre)
217 
218             NewWeed.Cost, Sigma, Delta = CalculateCost(NewWeed)
219             Weeds.append(NewWeed)
220             NoOfWeeds += 1
221             TotalWeeds += 1
222 
223             line = ["0",",",str(NewWeed.Phase2),",",Sigma,",",Delta,"\
                    n"]
224             F_input.writelines(line)
225     line = ["Iteration no:", ",", str(Iteration), "\n"]
226     F_input.writelines(line)
227     Iteration += 1 #Increment which iteration we're on
228 
229 F_input.close()
230 print "Total weeds tested: ", TotalWeeds
```

```
231 Weeds.sort(key = lambda x: x.Cost, reverse = True)
232 if SolutionFound == False:
233     print "I␣haven't␣reached␣the␣threshold␣but␣the␣best␣was", Weeds
          [0].Cost, "at", Weeds[0].Phase2
234
235 print datetime.now() - startTime
236 raw_input("Press␣enter␣to␣exit")
```

Appendix C

Flow diagrams used to describe program flow during HDL development

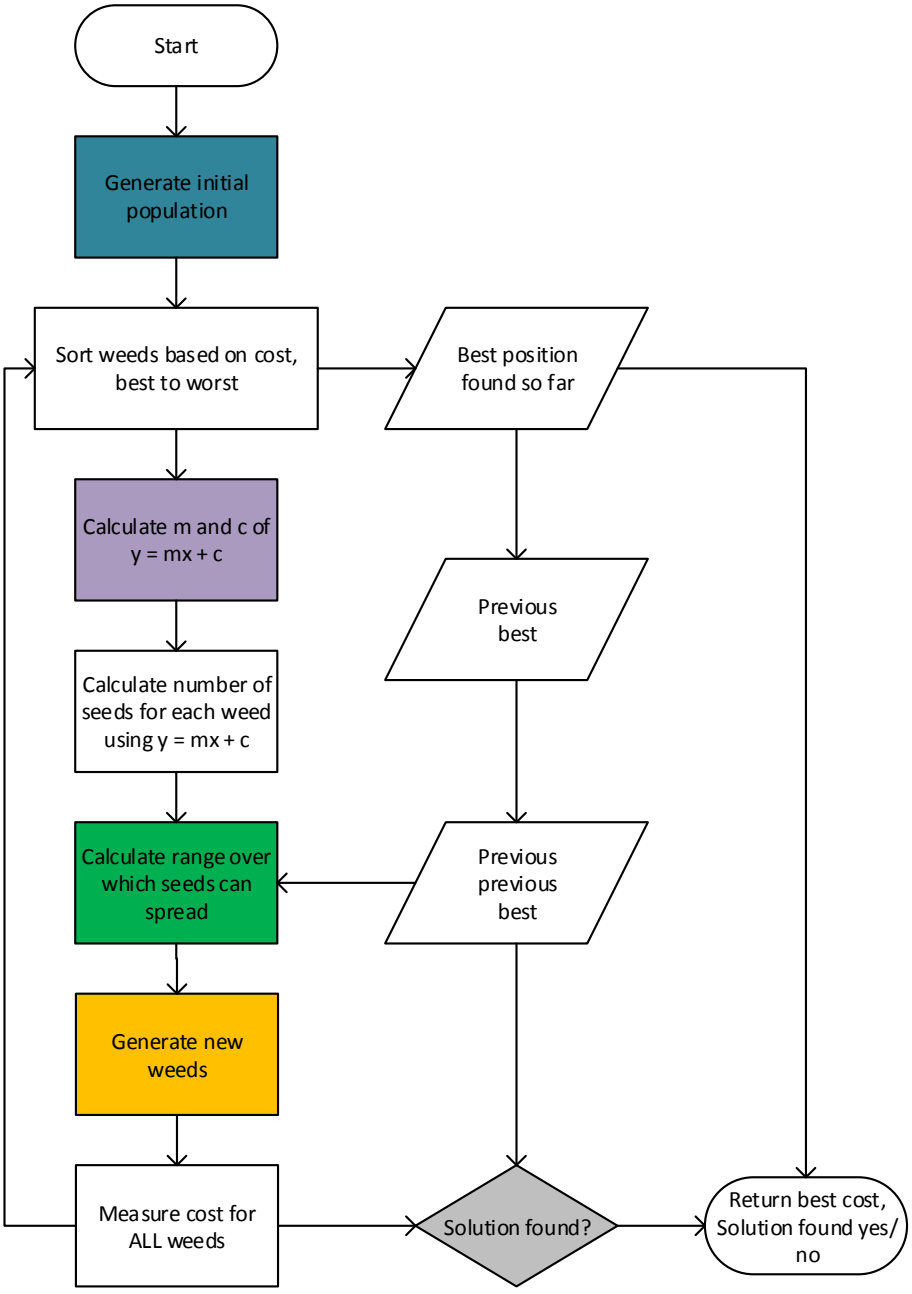


Figure C.1: Overall flow of the IWO algorithm.

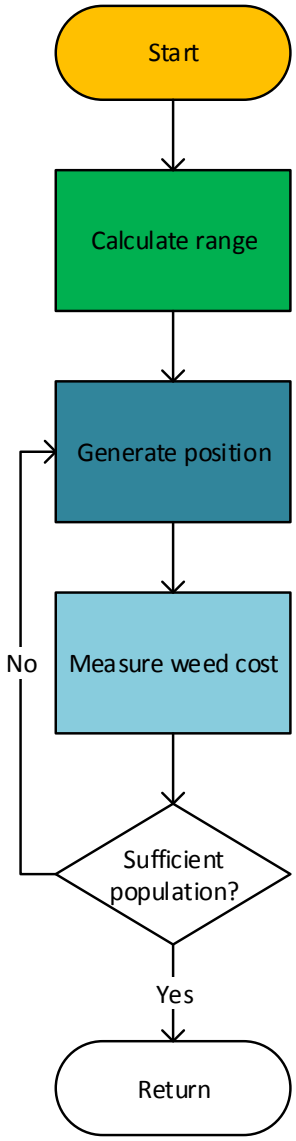


Figure C.2: Generating a new weed during the tracking IWO algorithm.

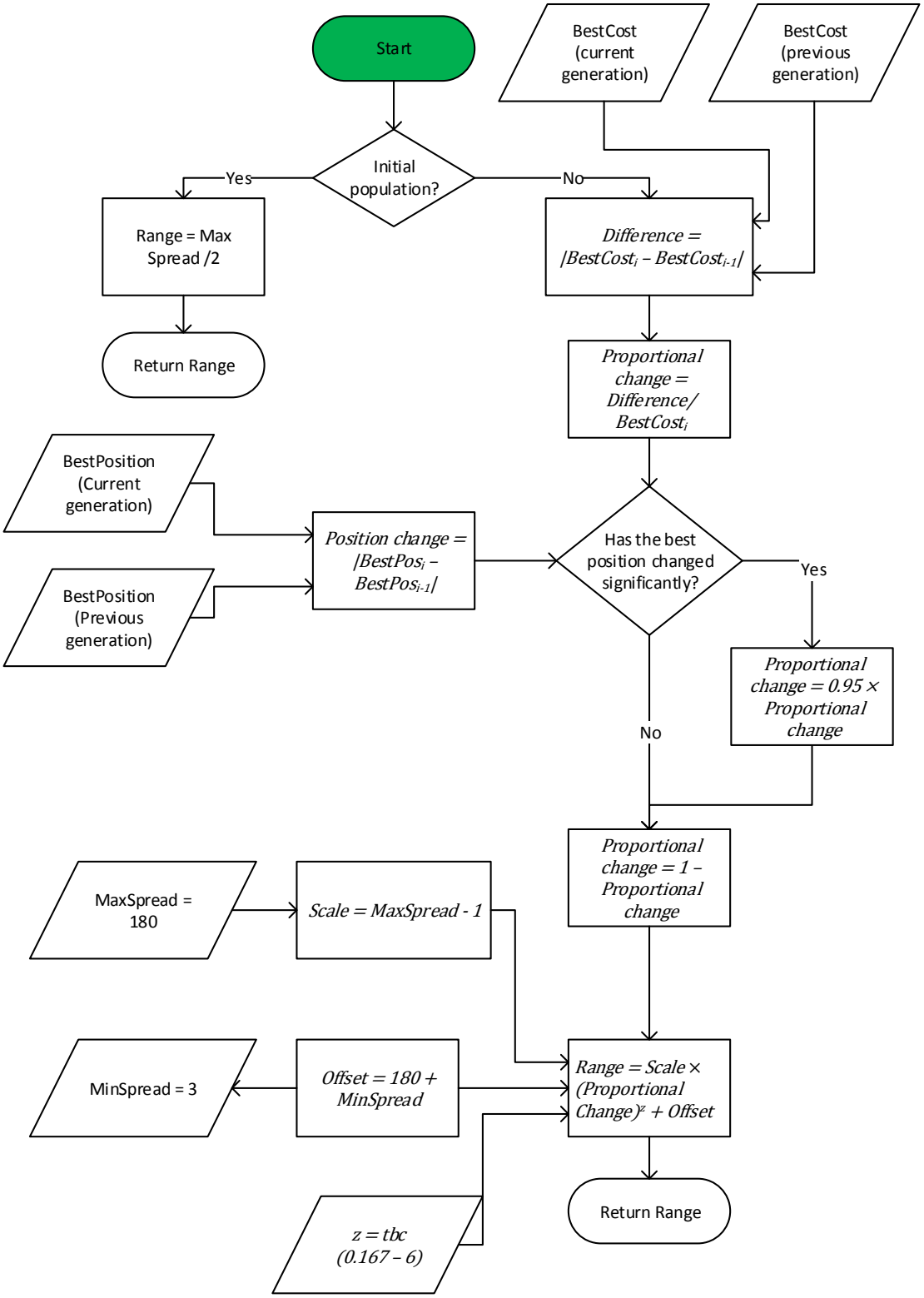


Figure C.3: Calculating the range during the tracking IWO algorithm.

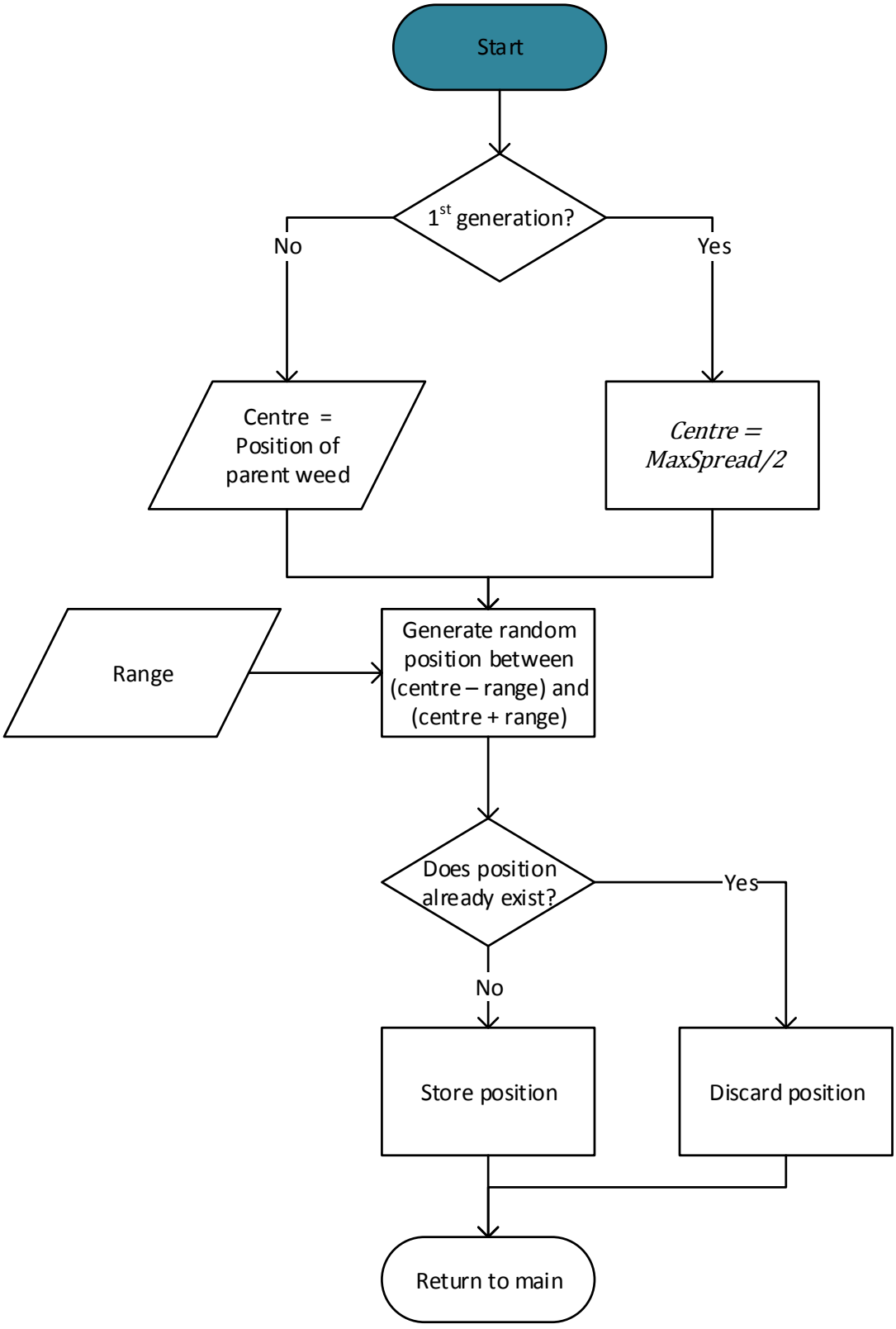


Figure C.4: Generating a position for a new weed during the tracking IWO algorithm.

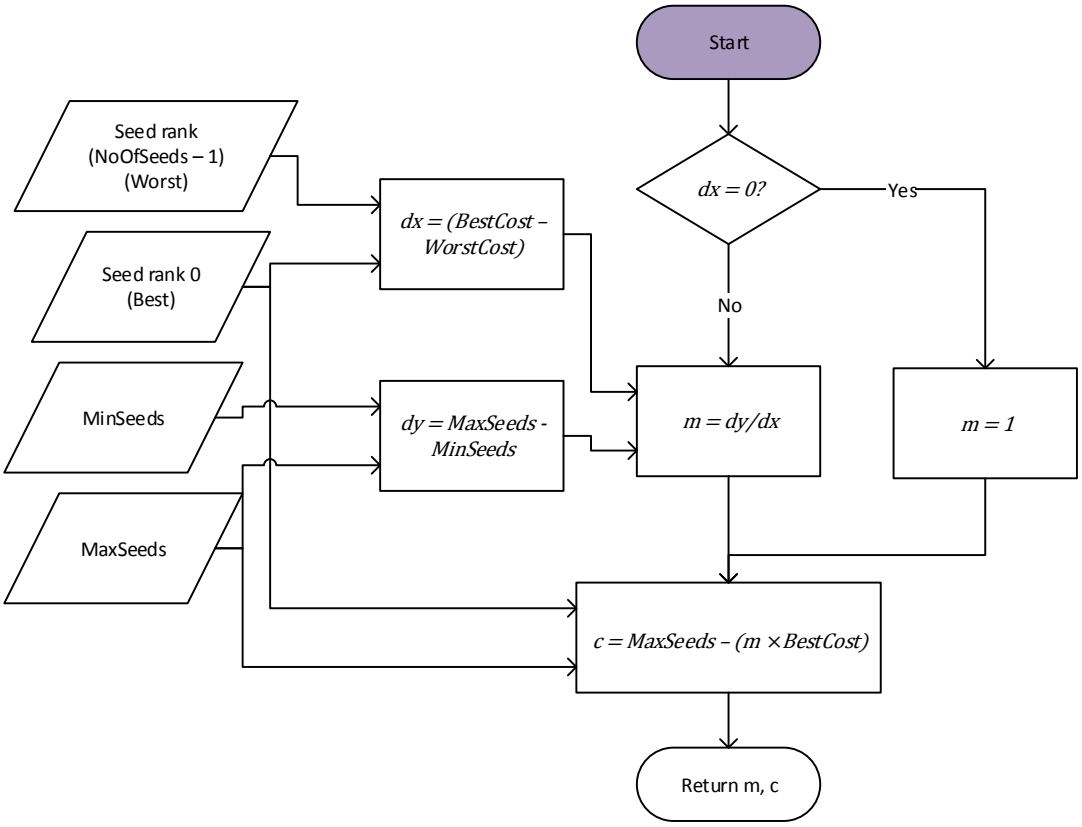


Figure C.5: Calculating m and c during the tracking IWO algorithm.

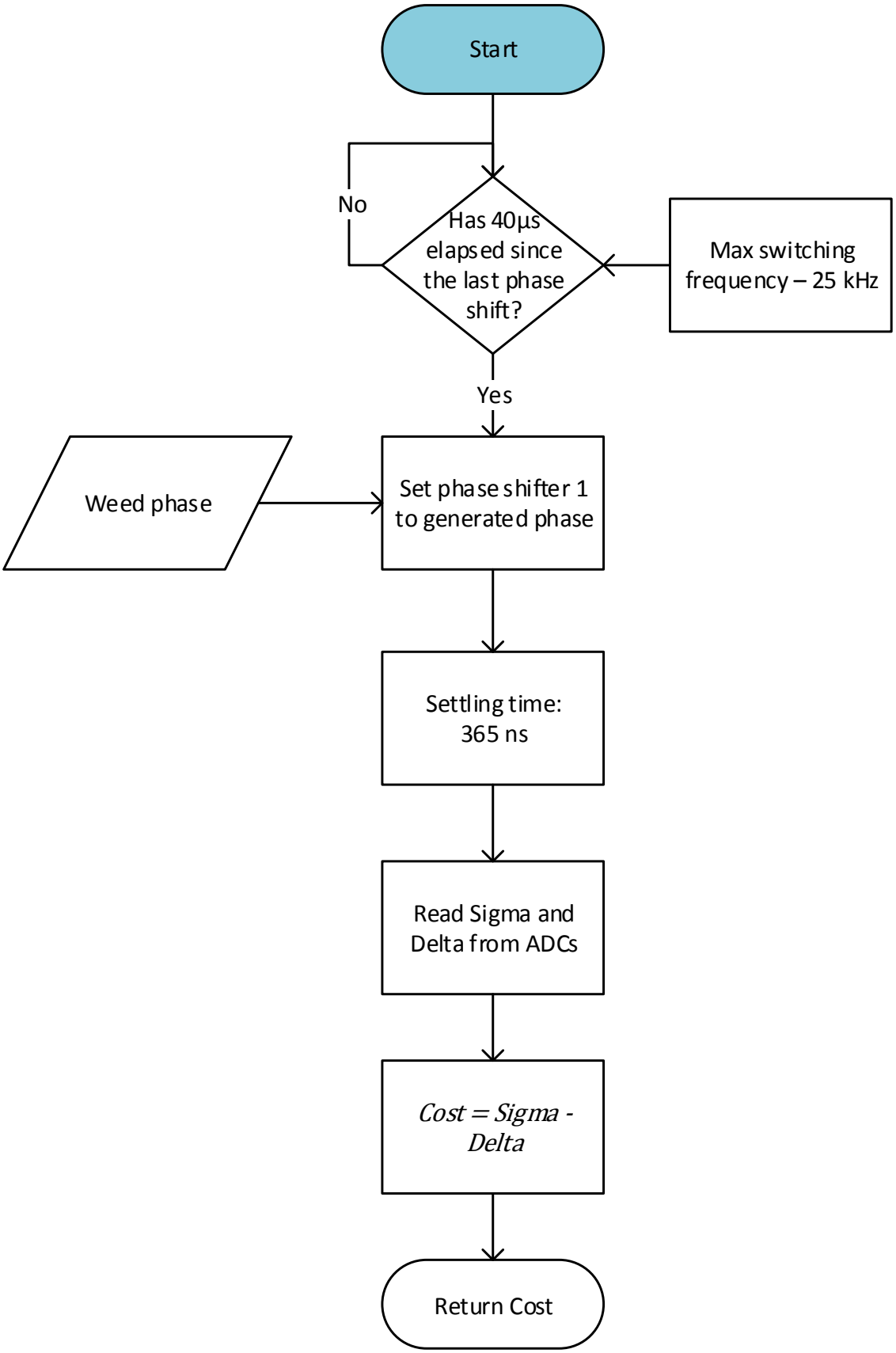


Figure C.6: Measuring the cost of a new weed during the tracking IWO algorithm.

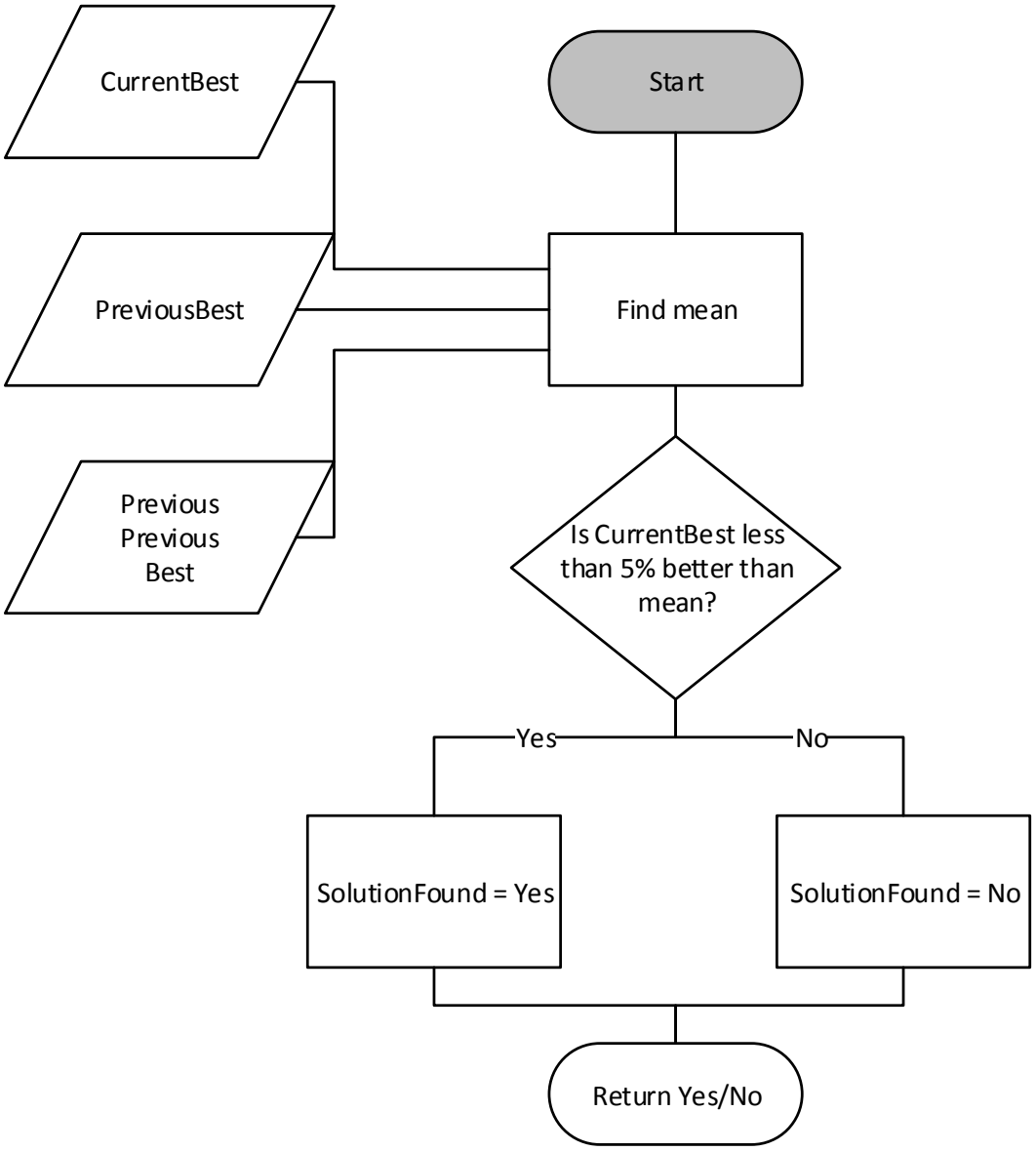


Figure C.7: Determining if a solution has been found during the tracking IWO algorithm.